The Journal of Chemical Physics

Recent developments in the general atomic and molecular electronic structure system

Cite as: J. Chem. Phys. **152**, 154102 (2020); https://doi.org/10.1063/5.0005188
Submitted: 20 February 2020 . Accepted: 19 March 2020 . Published Online: 16 April 2020

Giuseppe M. J. Barca, Colleen Bertoni, Laura Carrington, Dipayan Datta, Nuwan De Silva, J. Emiliano Deustua, Dmitri G. Fedorov, Jeffrey R. Gour, Anastasia O. Gunina, Emilie Guidez, Taylor Harville, Stephan Irle, Joe Ivanic, Karol Kowalski, Sarom S. Leang, Hui Li, Wei Li, Jesse J. Lutz, Ilias Magoulas, Joani Mato, Vladimir Mironov, Hiroya Nakata, Buu Q. Pham, Piotr Piecuch, David Poole, Spencer R. Pruitt, Alistair P. Rendell, Luke B. Roskop, Klaus Ruedenberg, Tosaporn Sattasathuchana, Michael W. Schmidt, Jun Shen, Lyudmila Slipchenko, Masha Sosonkina, Vaibhav Sundriyal, Ananta Tiwari, Jorge L. Galvez Vallejo, Bryce Westheimer, Marta Włoch, Peng Xu, Federico Zahariev, and Mark S. Gordon

The Journal of Chemical Physics ARTICLE scitation.org/journal/jcp

Recent developments in the general atomic and molecular electronic structure system

Cite as: J. Chem. Phys. 152, 154102 (2020); doi: 10.1063/5.0005188
Submitted: 20 February 2020 • Accepted: 19 March 2020 •
Published Online: 16 April 2020 • Corrected: 24 April 2020

Giuseppe M. J. Barca,[1] Colleen Bertoni,[2] Laura Carrington,[3] Dipayan Datta,[4] Nuwan De Silva,[5] J. Emiliano Deustua,[6] Dmitri G. Fedorov,[7] Jeffrey R. Gour,[8] Anastasia O. Gunina,[4] Emilie Guidez,[9] Taylor Harville,[4] Stephan Irle,[10] Joe Ivanic,[11] Karol Kowalski,[12] Sarom S. Leang,[3] Hui Li,[13] Wei Li,[14] Jesse J. Lutz,[15] Ilias Magoulas,[6] Joani Mato,[4] Vladimir Mironov,[16] Hiroya Nakata,[17] Buu Q. Pham,[4] Piotr Piecuch,[6,18] David Poole,[4] Spencer R. Pruitt,[4] Alistair P. Rendell,[1] Luke B. Roskop,[19] Klaus Ruedenberg,[4] Tosaporn Sattasathuchana,[4] Michael W. Schmidt,[4] Jun Shen,[6] Lyudmila Slipchenko,[20] Masha Sosonkina,[21] Vaibhav Sundriyal,[21] Ananta Tiwari,[3] Jorge L. Galvez Vallejo,[4] Bryce Westheimer,[4] Marta Włoch,[22] Peng Xu,[4] Federico Zahariev,[4] and Mark S. Gordon[4,a)]

AFFILIATIONS

[1] Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia
[2] Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont, Illinois 60439, USA
[3] EP Analytics, 12121 Scripps Summit Dr. Ste. 130, San Diego, California 92131, USA
[4] Department of Chemistry and Ames Laboratory, Iowa State University, Ames, Iowa 50011, USA
[5] Department of Physical and Biological Sciences, Western New England University, Springfield, Massachusetts 01119, USA
[6] Department of Chemistry, Michigan State University, East Lansing, Michigan 48824, USA
[7] Research Center for Computational Design of Advanced Functional Materials (CD-FMat), National Institute of Advanced Industrial Science and Technology (AIST), Umezono 1-1-1, Tsukuba 305-8568, Japan
[8] Microsoft, 15590 NE 31st St., Redmond, Washington 98052, USA
[9] Department of Chemistry, University of Colorado Denver, Denver, Colorado 80217, USA
[10] Computational Science and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830, USA
[11] Advanced Biomedical Computational Science, Frederick National Laboratory for Cancer Research, Frederick, Maryland 21702, USA
[12] Physical Sciences Division, Battelle, Pacific Northwest National Laboratory, K8-91, P.O. Box 999, Richland, Washington 99352, USA
[13] Department of Chemistry, University of Nebraska, Lincoln, Nebraska 68588, USA
[14] School of Chemistry and Chemical Engineering, Key Laboratory of Mesoscopic Chemistry of Ministry of Education, Institute of Theoretical and Computational Chemistry, Nanjing University, Nanjing 210023, People's Republic of China
[15] Center for Computing Research, Sandia National Laboratories, Albuquerque, New Mexico 87185, USA
[16] Department of Chemistry, Lomonosov Moscow State University, Leninskie Gory 1/3, Moscow 119991, Russian Federation
[17] Kyocera Corporation, Research Institute for Advanced Materials and Devices, 3-5-3 Hikaridai Seika-cho, Souraku-gun, Kyoto 619-0237, Japan
[18] Department of Physics and Astronomy, Michigan State University, East Lansing, Michigan 48824, USA
[19] Cray Inc., a Hewlett Packard Enterprise Company, 2131 Lindau Ln #1000, Bloomington, Minnesota 55425, USA
[20] Department of Chemistry, Purdue University, West Lafayette, Indiana 47907, USA
[21] Department of Computational Modeling and Simulation Engineering, Old Dominion University, Norfolk, Virginia 23529, USA
[22] 530 Charlesina Dr., Rochester, Michigan 48306, USA

Note: This article is part of the JCP Special Topic on Electronic Structure Software.
[a)] **Author to whom correspondence should be addressed:** mark@si.msg.chem.iastate.edu

ABSTRACT

A discussion of many of the recently implemented features of GAMESS (General Atomic and Molecular Electronic Structure System) and LibCChem (the C++ CPU/GPU library associated with GAMESS) is presented. These features include fragmentation methods such as the fragment molecular orbital, effective fragment potential and effective fragment molecular orbital methods, hybrid MPI/OpenMP approaches to Hartree–Fock, and resolution of the identity second order perturbation theory. Many new coupled cluster theory methods have been implemented in GAMESS, as have multiple levels of density functional/tight binding theory. The role of accelerators, especially graphical processing units, is discussed in the context of the new features of LibCChem, as it is the associated problem of power consumption as the power of computers increases dramatically. The process by which a complex program suite such as GAMESS is maintained and developed is considered. Future developments are briefly summarized.

 https://doi.org/10.1063/5.0005188

I. OVERVIEW/BACKGROUND

GAMESS (General Atomic and Molecular Electronic Structure System) was originally developed by Dupuis and co-workers in the late 1970s under the auspices of the National Resource for Computational Chemistry (NRCC), an organization that was sponsored by the National Science Foundation. GAMESS is a multi-functional electronic structure program with users in more than 100 countries and is run on nearly every available architecture, ranging from MacOS and Windows to the pre-exascale system Summit at Oak Ridge National Laboratory. GAMESS is a "cousin" of the HONDO program, which continues to be developed by Dupuis. GAMESS is distributed at no cost with a very simple license to prevent unauthorized redistribution. GAMESS itself is primarily written in Fortran77, with an increasing number of functionalities written in Fortran90. Associated with GAMESS is an object-oriented C++ library called LibCChem, initiated in 2010, which contains an increasing number of quantum chemistry functionalities and is written for both central processing unit (CPU) and GPU (graphical processing unit) architectures.

As discussed in two previous reviews in 1993[1] and 2005,[2] GAMESS has essentially all of the commonly used electronic structure methods, including Hartree–Fock (HF) self-consistent field (SCF), density functional theory (DFT) with many of the popular functionals, second order perturbation theory (MP2), coupled cluster (CC) theory, including CCSD(T), and novel methods such as CR-CC(2,3) that are capable of correctly breaking single bonds, equations-of-motion (EOM) coupled cluster theory, time-dependent density functional theory (TDDFT), configuration interaction (CI) up to and including full CI, complete active space (CAS) SCF, multi-reference (MR) MP2, and multi-reference CI (MRCI). Also available in GAMESS is the effective fragment potential (EFP) method, a sophisticated model potential with no fitted parameters, which is applicable to any molecular system. Other functionalities include fully analytic second energy derivatives (Hessians) for closed shell HF and CASSCF, fully analytic energy first derivatives (gradients), and, therefore, semi-numeric Hessians for HF, DFT, MP2, CI, and EFP, thereby enabling the prediction of vibrational frequencies and IR and Raman spectra. Related to vibrational spectroscopy is the vibrational SCF suite of methods developed by Gerber and coworkers.[3] GAMESS also has several options for reaction path following and for performing classical trajectories using any of the available electronic structure methods. Solvent effects can be incorporated explicitly using the EFP method or implicitly using the polarizable continuum model (PCM[4]), COSMO (conductor-like Screening Model),[5] or the surface volume polarization (SVP) model.[6] Surface science can be studied using the surface integrated molecular orbital molecular mechanics (SIMOMM)[7] method.

If one desires very high accuracy in electronic structure calculations, there is a CEEIS (correlation energy extrapolation by intrinsic scaling)[8] method developed by Ruedenberg and Bytautas that provides essentially the exact full CI energy at a fraction of the cost.

The ability of GAMESS to treat excited electronic states, photochemistry, and related phenomena such as surface crossings and conical intersections has made significant advances with the introduction of spin-flip (SF) methods[9] for the energy and the analytic gradient,[10] including the development of a general approach to spin-correct spin-flip.[11]

An exciting new feature of GAMESS is the quasiatomic orbital (QUAO) analysis developed by West and colleagues.[12] This analysis, which continues to be developed, has been applied to several interesting problems in chemistry.

Since the early 1990s, a major effort related to the development of GAMESS has been to maximize the scalability (parallelism) of the code. The ability of GAMESS to explore potential energy surfaces accurately and efficiently is much improved with the development of several GAMESS functionalities that can take advantage of combining MPI (message passing interface) and OpenMP into a hybrid approach that takes optimal advantage of both distributed computing (MPI) and shared memory computing (OpenMP). This combination has now been applied to HF, DFT, and the resolution of the identity (RI) version of MP2.

In the past several years, this stride toward high performance computational chemistry has increasingly taken center stage.[13–18] An important component of this endeavor has been to make optimal use of accelerators. In the remainder of this review, the primary focus is on new features that have been implemented since 2005 and, in particular, the advances in the development of highly scalable code, with the aim of achieving the ability to make use of the anticipated exascale computers, where exascale may be defined as 10^{18} flops or a gigagigaflop.

An important component of the development of highly scalable electronic structure software is the innovation of reliable fragmentation methods. In GAMESS, this specifically means the fragment molecular orbital (FMO),[19] the effective fragment potential (EFP),[20] and the effective fragment molecular orbital (EFMO)[21] methods.

Together, these methods facilitate the capability to address problems that contain tens of thousands of atoms with an accuracy that is equivalent to that of correlated electronic structure methods. Another type of fragmentation subdivides wave functions, rather than physical atoms or groups of atoms. Two such methods are ORMAS[22] (occupation restricted multiple active spaces) and CIM (cluster in molecule).[23] The ORMAS method has been enhanced by the addition of dynamic correlation via second order perturbation theory (ORMAS-PT2),[24] thereby enabling accurate studies of excited electronic states. The ORMAS method also enabled the development, mentioned above, of a general spin-correct spin flip method. The CIM method, developed by the Piecuch group, has been combined with the FMO method[25] to enable fast and accurate coupled cluster calculations.

A second key component of the stride toward exascale computing is the recognition that accelerators/co-processors, such as GPUs, will play an important role in the future of high performance computational chemistry. In the last decade, this recognition led to the development of the C++ CPU/GPU library LibCChem that is attached to GAMESS and has an expanding array of functionalities. These and other new GAMESS developments will be discussed in Secs. II–V.

II. FRAGMENTATION METHODS

The development of fragmentation methods in GAMESS has played a central role in the advance toward massively parallel computing capability, since each fragment can be computed essentially independently of every other fragment. This means that the computational bottleneck reduces from that of the entire molecular system to that of the largest fragment. In the following, several fragmentation methods that are available in GAMESS are discussed.

A. Fragment molecular orbital theory

The FMO method[26] was first released in GAMESS in 2004.[27] FMO is a QM approach capable of evaluating the properties of large molecular systems;[28–30] the largest system computed with FMO2/DFTB in GAMESS has about 1.2×10^6 atoms.[31] To enable this large-scale molecular dynamics (MD) simulation, considerable efforts were invested in improving the MD engine in GAMESS.

FMO in GAMESS is efficiently parallelized using the multilayer hierarchical parallelization scheme, generalized distributed data interface (GDDI)[27] possibly in combination with OpenMP.[32] Good parallel efficiency was reported for FMO simulations on supercomputers using GDDI.[33] GDDI can also be used for parallelization of non-FMO simulations, provided that they have some granularity in terms of tasks, for instance, different replicas in replica exchange MD.[34,35] Various properties can be computed with FMO in GAMESS, as summarized in Table I.

FMO can be combined with many but not all QM methods available in GAMESS. The QM methods interfaced with FMO are listed in Table II. In order to compute the analytic gradient for FMO accurately, it is necessary to evaluate orbital responses (derivatives of molecular orbital coefficients with respect to nuclear coordinates) by solving coupled-perturbed Hartree–Fock (CPHF) equations. This can be done efficiently using the self-consistent Z-vector method (SCZV),[36] which has to be formulated for each wave function separately, and not all QM methods can be used with SCZV at present. Analytic second derivatives can be evaluated in FMO by solving a different set of CPHF equations. Among all methods, HF and DFT have been most extensively extended to treat open-shell systems.

The FMO method has also been interfaced with the cluster-in-molecule (CIM) method developed by Piecuch and co-workers, as discussed in Sec. III B. CIM is based on an orbital partitioning, rather than a physical partitioning of atoms using localized molecular orbitals (LMOs). The main CIM bottleneck is the need to localize the orbitals of the entire system, no matter how large. In analogy with the FMO method itself, the FMO/CIM method[25] reduces the bottleneck to the localization of the orbitals of the largest fragment.

B. Effective fragment potential

The effective fragment potential (EFP) method is an *ab initio* force field designed to model intermolecular interactions accurately and efficiently. In EFP, parameters for each individual fragment (monomer) are generated from a single point *ab initio* calculation, typically at the HF level for a chosen geometry (the MAKEFP run described below). In an EFP calculation, all fragments are internally rigid, i.e., they have a fixed geometry. The interaction energy between EFP fragments (EFP–EFP) and the interaction energy between EFP fragments and a molecule described by a quantum mechanical (QM) wave function, if one is present (QM–EFP), are computed. The QM–EFP approach was implemented in order to

TABLE I. Properties that can be computed with FMO in GAMESS.

Property	Reference
Harmonic frequencies, IR and Raman spectra	37
Electronic excitations	38
Electron density and molecular electrostatic potential on a grid	39
MOs, their energies, and density of states	40
Minimum energy crossing point of energy surfaces	41
Interaction energy analysis for explicit solvent	42
Pair interaction energy decomposition analysis for implicit solvent	43
Fluctuation analysis of pair interactions in MD	44

TABLE II. The highest analytic derivative of the energy with respect to nuclear coordinates for each QM method interfaced with FMO in GAMESS (0 = energy, 1 = gradient, and 2 = Hessian).[a]

	Restricted closed shell	Restricted open shell	Unrestricted	Multi-reference
DFTB	2[45]			
HF	2[46]	2[47]	2[48]	1[49]
CIS	0[50]			
MP2	1[51]	1[47]	0[52]	
CC	0[53]	0[54]		
DFT	2[55]		2	
TDDFT	1[56]		0[38]	
PCM[b]	2[37]	1[52]	1[52]	

[a]It is possible to combine several methods in the multilayer approach.[57] DFTB = density functional tight binding, CIS=CI singles, and CC = coupled cluster.
[b]Polarizable continuum model (PCM) can be combined with other QM methods; it comes with its own limitations in the current implementation, as shown in the table; for example, the highest derivative for RMP2/PCM is min(1,2) = 1.

handle the situation in which significant changes occur in the geometry or electronic structure of the QM region, while the "spectator" molecules (EFP fragments) remain internally intact. It is worth noting that there are differences in the formulation of EFP–EFP and QM–EFP interaction components though they arise from the same theory.

The original EFP method (called EFP1) was designed to model aqueous solvation only. All parameters in EFP1 are stored within GAMESS and do not need to be generated from a MAKEFP calculation. The EFP1–EFP1 and QM–EFP1 interaction energies are composed of three terms,

$$E = E_{Coulomb} + E_{polarization} + E_{remainder}. \quad (1)$$

The first term, $E_{Coulomb}$, is the Coulomb interaction energy between distributed multipoles of different fragments located at atom centers and bond midpoints, generated using the distributed multipole analysis by Stone.[58] $E_{polarization}$ is the polarization energy computed by iteratively converging the induced dipole moments of the localized molecular orbitals (LMOs) to self-consistency. $E_{remainder}$ is the remainder interaction energy. This term is fitted to reproduce the HF[59] or DFT (B3LYP) interaction energy[60] of the water dimer at various points of the potential energy surface. For the HF derived parameters, the remainder term includes exchange–repulsion and charge transfer. For the DFT derived remainder term, electron correlation from the B3LYP functional is also included in the interaction energy.

For QM–EFP1, the Coulombic effect from the distributed multipoles of EFP fragments is included as a perturbation to the QM one-electron Hamiltonian.[59] Also contributing to the QM one-electron Hamiltonian is the polarization between the charge density of the *ab initio* region and the induced dipoles of EFP fragments, both of which are converged to self-consistency.[59] The QM–EFP and EFP–EFP polarization are non-separable because the induced dipoles of one fragment depend on the static multipoles and induced dipoles of all other EFP fragments, as well as the charge density of the *ab initio* region. The remainder term also affects the one-electron Hamiltonian of the QM part.

QM–EFP1 has been shown to successfully describe aqueous solvent effects for both ground and excited electronic state properties and processes.[59,61–68] Several types of methods can be used to describe the QM region, including Hartree–Fock,[59] DFT, time-dependent DFT,[61] CIS,[62] MCSCF,[63] MP2, multi-reference MP2,[64] coupled cluster (CC), and the equation-of-motion CC (EOM-CC) suite.[66] QM–EFP1 has been interfaced with the polarizable continuum model (PCM).[4] A recent development for the EFP1 method is the addition of a dispersion energy term. Both an empirical dispersion term[69] and the first principles derived dispersion term[70] were implemented, as described below. All EFP1–EFP1 and QM–EFP1 analytic gradients have been derived and implemented.[59] Therefore, one can perform geometry optimizations and molecular dynamics simulations.

Recently, De Silva, Adreance, and Gordon implemented the Grimme–D3 semi-empirical dispersion energy correction (including the "E_8 term") for EFP1 and for QM–EFP1 systems.[69] The resulting method is called EFP1–D3, or QM–EFP1–D3 if there is a QM component. Since the D3 correction can be computed with force field speed, the computational cost of this method is trivial. In addition, the EFP1 and QM–EFP1 analytic gradients with the D3 correction have been developed and are available in GAMESS, thereby enabling the geometry optimizations of water clusters and solute–water complexes, with the dispersion effect included. This method has been applied to a broad range of test molecules: neutral water clusters, protonated and deprotonated water clusters, and auto-ionized water clusters (water27[71] test set), as well as solute–water binary complexes (all of the water-containing complexes in the S66[72] test set). The EFP1–D3 and QM–EFP1–D3 binding energies of the above test molecules are in good agreement with those obtained using MP2 and CCSD(T) at the complete basis set (CBS) limit. The binding energies are considerably improved (errors are reduced by roughly half) compared to EFP1 and QM–EFP1 without the dispersion correction. The EFP1–D3 and QM–EFP1–D3 methods are important for evaluating the molecular properties of large water and water–solute molecular systems for which the computational cost can be significant otherwise.

Another dispersion correction to EFP1, derived from first principles, was implemented.[70] This dispersion energy term is identical to the one used in the EFP2 method and is currently implemented only for EFP–EFP interactions. The parameters needed to compute this energy term are generated at the RHF/DH(d,p) level of theory, similar to the other parameters in EFP1.

Fitting to *ab initio* potentials for every species of interest is neither desirable nor practical. The EFP1 method was later extended to EFP2 to model any (closed shell) molecule. In EFP2, the intermolecular interaction energy is given by

$$E = E_{Coulomb} + E_{polarization} + E_{dispersion} + E_{exchange\text{-}repulsion} + E_{charge\ transfer}, \quad (2)$$

where $E_{Coulomb}$ and $E_{polarization}$ are defined in the same manner as in EFP1. The dispersion energy $E_{dispersion}$ is computed with LMO polarizability tensors. The exchange–repulsion energy, $E_{exchange\text{-}repulsion}$, arising from the Pauli repulsion, is derived from a power expansion of the intermolecular overlap.[73] The charge transfer energy, $E_{charge\ transfer}$, is the stabilizing interaction between occupied MOs of one fragment and unoccupied MOs of another.[74] In contrast to EFP1, the EFP2 parameters are all generated from first principles without any empirical fitting. A recent addition to the EFP2–EFP2 interaction energy is the R^{-7} dispersion interaction.

For QM–EFP2, the Coulomb and polarization terms are the same as in QM–EFP1. The remainder term of QM–EFP1 is replaced by explicit formulations of the dispersion and exchange–repulsion terms in QM–EFP2. The effect of exchange–repulsion is accounted for via the exchange–repulsion Fock contribution to the one-electron part of QM Hamiltonian, whereas the dispersion energy is added as a post-SCF energy correction. Unlike the Coulomb and polarization terms, the QM–EFP2 exchange–repulsion term[75] contains explicit electron repulsion integrals (ERIs), making it the most expensive term in the QM–EFP2 method. The QM–EFP2 R^{-6} dispersion coefficients are computed using EFP LMO dynamic dipole polarizabilities and the dipole integrals as well as the orbital energies of the QM part.[76] Currently, the QM–EFP2 R^{-7} dispersion component and the charge transfer term are not yet implemented. Recent developments for QM–EFP2 will be described below.

When two molecules are sufficiently close and their electron density overlap is large, the multipole approximation becomes inadequate due to its classical nature. Damping/screening functions must be introduced for the Coulomb interaction to ensure the correct asymptotic behavior. Similarly, polarization and dispersion, which are developed from intermolecular perturbation theory based on the negligible overlap assumption, also demand proper screening at short-range. For the EFP–EFP interactions, several damping functions for the Coulomb, polarization, and dispersion interactions are implemented.[77] After the development of the R^{-7} dispersion interaction, the overlap-based dispersion damping function was reformulated to incorporate odd-power terms.[78] For QM–EFP2, currently, the Coulomb interaction employs Gaussian damping and the dispersion interaction can be screened using either Tang–Toennies or overlap-based damping functions.[76,79] The recent developments for damping functions of different interaction terms will be mentioned below.

Since EFP2 is an *ab initio* force field, the MAKEFP run is employed to generate all parameters from first principles. These parameters essentially comprise various properties of a fragment, computed at the HF level of theory. The computed EFP2 parameters can be either printed out to a file (with .efp extension) and subsequently inserted into an EFP2–EFP2 or QM–EFP2 job input, possibly through a library of standard fragments, or computed on-the-fly for the Effective Fragment Molecular Orbital (EFMO) calculations (discussed in Sec. II C).

The five terms in the EFP2–EFP2 energy expression [Eq. (2)] need the following input for each type of fragment: The Coulomb term requires multipole moments, distributed over atomic centers and bond midpoints. The polarization term requires polarizability tensors, distributed over LMO centroids. The dispersion term needs distributed dynamic polarizability tensors, again, over LMO centroids. The exchange–repulsion term utilizes data on fragment LMOs, while the charge transfer contribution uses either canonical molecular orbitals (CMOs), already computed within the HF calculation, or the valence virtual orbitals (VVOs)[80] for a more computationally efficient truncated virtual space. The screening parameters for the Coulomb term are computed by fitting on a grid the damped classical multipolar electrostatic potential to the quantum potential of the fragment, with damping functions having either a Gaussian or an exponential form.[77,81]

Like much of GAMESS, the EFP method and the MAKEFP module evolve over time. Because the MAKEFP calculation to establish the EFP parameters is a significant bottleneck, considerable effort has been expended to make EFP more computationally efficient and scalable. The parallelization approach for MAKEFP is shared memory, motivated by the opportunity to use a hybrid MPI/OpenMP parallel approach for the EFMO method. The parallelization is done using OpenMP pragmas. The distributed nature of the EFP2 potential and the manner in which some of the EFP parameters are structured provide parallelization opportunities. For instance, in addition to being distributed, the parameters for the Coulomb term have independent orders of multipole moments and the dispersion term requires 12 independent frequencies. The screening parameters for the Coulomb term are computed on a grid, which is inherently parallelizable. There are other points of parallelization within the MAKEFP code as well. An additional level of performance improvement for the code is achieved via minimizing I/O within the MAKEFP workflow, except printing to the .efp file (EFMO uses in-memory data transfer for the EFP parameters).

The dispersion interaction is often expressed as an expansion of inverse powers of distances between relevant molecular moieties,

$$E^{disp} = \sum_{n\geq 6} \frac{C_n}{R^n}, \quad (3)$$

where R represents the distances between molecular moieties and n starts at 6, which represents the induced dipole-induced dipole part of the dispersion interaction. Most methodologies that treat dispersion do not include the odd-power terms and simply fit the C_6 coefficient to experimental or to high-level *ab initio* values. Such an approach can work because the fitted parameters can cover up deficiencies in the underlying potential. To better understand the effect of the odd-power dispersion terms, the leading odd-power term, R^{-7} dispersion (Disp7), was implemented for the EFP2–EFP2 dispersion

interaction, utilizing the frequency-dependent anisotropic Cartesian polarizabilities located at the centroids of the LMOs of the EFP fragments. It was shown that Disp7, although it can rotationally average to zero in some situations, can be either attractive or repulsive, with substantial magnitudes relative to the R^{-6} dispersion, and is highly dependent on the orientation of the molecules.[82] Furthermore, a benchmarking study based on the S22 dataset has demonstrated that in hydrogen-bonded systems, Disp7 almost always is repulsive and has a substantial magnitude (as large as 50% of the R^{-6} term in some cases), whereas it makes an insignificant contribution for other types of complexes.[83] The analytic gradient for Disp7 has been derived and implemented, which allows one to take the Disp7 effect into account for geometry optimizations and molecular dynamics simulations.[78] In addition, the overlap-based damping function, which was originally developed only for the even-power terms, now has been reformulated to incorporate the odd-power terms.[78]

Recently, the QM–EFP2 method has been reassessed, and several improvements were made to the Coulomb, exchange–repulsion, and dispersion terms. It was realized that, unlike QM–EFP1 where both the nuclear and electronic charges of EFP fragments were damped, the Gaussian damping function in the Coulomb term was only applied to the EFP electronic charges in QM–EFP2. This seemingly insubstantial difference led to large discrepancies of the QM–EFP2 Coulomb energy compared to either EFP2–EFP2 or symmetry adapted perturbation theory (SAPT).[84,85] Now, both QM–EFP1 and QM–EFP2 Gaussian damping functions screen all of the EFP effective charges (nuclear + electronic).[21]

For the exchange–repulsion term, the spherical Gaussian overlap (SGO) approximation, which provides accurate EFP2–EFP2 energies with high computational efficiency, was shown to cause large errors in the QM–EFP2 exchange–repulsion energy. Hence, the recently revised formulation completely removes the SGO approximation and computes the ERIs explicitly. Moreover, the early QM–EFP2 exchange–repulsion implementation was limited to only one EFP fragment. The current implementation has been successfully tested for water clusters with hundreds to thousands of fragments. The new implementation is dramatically improved by employing the direct (on-the-fly) approach for computing ERIs with either pure MPI or hybrid MPI/OpenMP parallelization schemes, in contrast to the original disk-based serial implementation.[79]

As was done for the EFP2–EFP2 dispersion, the QM–EFP2 dispersion energy needs to be screened to ensure the correct asymptotic behavior. Both the Tang–Toennies and the overlap-based damping functions are available to account for exchange–dispersion and charge penetration effects at short-range. Very recently, the overlap-based damping formula has been updated to be of the same functional form as the EFP2–EFP2 overlap-based damping function.[79]

Currently, QM–EFP2 has been coupled with HF, DFT, MP2, and CC. The development of QM–EFP2 gradients is in progress. All of these efforts will allow better prediction and understanding of chemical properties in both ground and excited states in clusters and in the condensed phase.

C. Effective fragment molecular orbital method

The EFMO[86] method is a fragmentation method in a similar spirit to FMO. It combines the fragmentation scheme from FMO with the *ab initio* force field EFP method to account for the long-range and many-body terms. The method was developed to take advantage of the computational efficiency of both methods so that computations on molecules that were previously out of reach for chemists due to the computational cost would become feasible. The initial version of EFMO was developed with only the Coulomb interaction and polarization terms from EFP included, but in Ref. 21, the remaining three terms in EFP (dispersion, exchange–repulsion, and charge transfer) were included. In Ref. 87, the fully analytic gradient for EFMO with the Coulomb, polarization, exchange–repulsion, and dispersion terms was reported. The analytic gradient for the charge transfer term is under development.

Similar to the FMO energy equation, the EFMO energy expression is a fragmentation-based many-body expansion, where the system is first divided into fragments (monomers). In EFMO, the energy is the sum of the monomer, dimer, and many-body polarization terms. The EFMO energy equation can be written as [Eq. (3.22) from Ref. 87]

$$E^{EFMO} = \sum_{A}^{fragments} E_A^0 + \sum_{A>B}^{R_{A,B}\le R_{cut}} \left(\Delta E_{AB}^0 - E_{AB}^{pol}\right) + \sum_{A>B}^{R_{A,B}>R_{cut}} E_{AB}^{EFP} + E_{tot}^{pol}, \tag{4}$$

where E_A^0 is the gas phase energy of fragment A, $E_{AB}^0 = E_{AB}^0 - E_A^0 - E_B^0$ (the dimer two-body interaction energy), and E_{AB}^{EFP} is the long-range EFP energy between fragments A and B

E_{tot}^{pol} is the EFP polarization energy for the entire system; E_{AB}^{pol} is the EFP polarization energy for fragments A and B; and $R_{A,B} = \min_{I\in A, J\in B} \frac{|\boldsymbol{r}_I - \boldsymbol{r}_J|}{V_I + V_J}$ is the relative interatomic distance between fragments A and B, where atoms I (J) are on fragment A (B) and V_I and V_J are the van der Waals radii of atoms I and J, respectively.

To compute the EFMO energy, first an *ab initio* method is chosen for the gas phase energy computations (e.g., RHF). Then, the monomer energy is computed by summing the gas phase energy for each monomer. Next, the dimer interaction energy is computed either using the chosen *ab initio* method or using the long-range EFP interaction energy as an approximation to the exact dimer interaction energy. To determine what is "long-range" and what is not, the relative distance between the fragments ($R_{A,B}$) is computed and compared to a user-supplied cutoff value R_{cut}. If the distance is larger than the cutoff, the fragment–fragment interactions are considered "long-range," and the EFP interaction energy is used. Finally, to account for many-body polarization effects, the EFP polarization energy between all fragments in the system is added to the energy.

The monomer and dimer terms in the EFMO method are different from those in the FMO method. Specifically, in EFMO, the monomer and dimer terms do not include the monomer Coulomb field. Instead, the EFMO method includes a many-body polarization term computed from all of the fragments.

The EFMO gradient can be computed by considering each term in Eq. (4). Each term in Eq. (4) is differentiated with respect to the x-coordinate of each atom K (x_K). Note that here E_{AB}^{EFP} is expanded to Coulomb, exchange–repulsion, dispersion, and charge transfer terms. The gradients of the *ab initio* energy terms can be computed with standard methods.[88] The gradients of the EFP terms except for charge transfer are discussed in Ref. 87. The main difference between the standard EFP gradients and the gradients of the EFP terms in

the EFMO gradient is that in standard EFP computations, EFP fragments are rigid, while in EFMO, the fragments are flexible. Taking into account flexible fragments results in additional response terms with response equations that need to be solved to compute a fully analytic gradient.

The anticipated US exascale computers are currently all planned to be heterogeneous systems in which each node contains multiple GPUs. To take advantage of the massive parallelism available, being able to effectively decompose the computation so that it can run in parallel across nodes as well as use the parallelism inside nodes is important.

Transitioning to exascale computing involves multi-grain, massive yet flexible parallelization of a code, adoption of accelerators, careful use of bandwidth, and memory structures. The structure of the EFMO method maps very naturally onto these requirements. As a fragmentation method, EFMO can have its independent monomer and dimer terms in energy and gradient expressions, mapped onto different nodes or sets of nodes of a supercomputer while also reducing memory requirements down to that for a fragment or a dimer. There are a limited number of communication points throughout the run (transitioning from monomers to dimers, reduction for computing total energy or gradient), and the only term with significant communication requirements is the total polarization, E_{tot}^{pol} and $\partial E_{tot}^{pol}/\partial x_K$, which is done once per single point energy calculation and then once per gradient point, respectively. Within other terms for each monomer and dimer, one can either use shared-memory parallelization or request a hybrid-parallel run at several nodes, depending on the scaling and implementation of the electronic structure method of choice and available computational resources. The electronic structure method can further utilize the offloading capability, if one is already implemented (see Sec. III A). Finally, describing many-body contributions via the polarization term allows one to stop at the dimers in the EFMO energy expression, reducing the scaling and memory bottlenecks to the requirements of the largest *ab initio* dimer.

III. ELECTRONIC STRUCTURE METHODS IN GAMESS

There have been many new electronic structure methods implemented in GAMESS in the last 15 years. These include novel implementations of the resolution of the identity (RI)-MP2 method, a multitude of coupled cluster methods thanks to the efforts of the Piecuch group, the ORMAS MCSCF and CI method including a second order perturbation theory correction, a coupled electron pair approximation (CEPA) suite of methods, the nearly exact correlation energy extrapolation with intrinsic scaling (CEEIS) method spin-correct spin flip methods based on ORMAS, the fundamental analysis of the chemical bond based on quasi-atomic orbitals (QUAOs), the density functional theory/tight binding (DFTB) method, and many new functionals mostly due to the Truhlar group. Each of these is discussed in Secs. III A–III K.

A. Hartree–Fock and second order perturbation theory using a hybrid MPI/OpenMP approach to parallel code

The introduction of the hybrid MPI/OpenMP parallel programming model to GAMESS is one of the efforts to design efficient and scalable electronic structure codes that can treat macromolecular systems at the *ab initio* level of accuracy. The combined MPI/OpenMP model has been used in GAMESS for both regular quantum mechanics (QM) methods and QM methods in the fragmentation context. In GAMESS, MPI is wrapped in the distributed data interface (DDI)[89] or the Generalized DDI (GDDI)[19] interface to assist distributed arrays allocated across multiple compute nodes and the multilevel parallelism using the MPI group concept. In this section, MPI mostly refers to the GDDI interface supporting the multilevel parallelism in fragmentation methods. By using the group concept, the GDDI arranges MPI compute processes (ranks) into groups. Ranks in the same group can communicate with each other referring to the same MPI communicator. This allows each group of ranks to work on independent chunks (e.g., a fragment *ab initio* calculation) that subsequently increases the parallel coverage and the scalability of the parallel code. In fact, the distributed memory model supported by the pure MPI model remains the best way to build and maintain very large scalable supercomputers. However, the pure MPI parallel model is known to suffer from a large memory footprint (e.g., due to replicated data in all ranks) and a high communication overhead (e.g., for its send/receive message protocol) in large scale calculations. This drawback becomes serious for the new multicore CPU generation. For instance, the Intel KNL compute node can have 64, 68, and 72 cores; each core has four threads, i.e., full CPU utilization on each compute node can support up to 256–288 compute processes. The MPI codes can rarely make use of more than half of these CPU cores.

Therefore, the hybrid MPI/OpenMP model was introduced to GAMESS to maintain the MPI scalability and boost the efficiency of the computation (e.g., by alleviating MPI restrictions). For the MPI/OpenMP fragmentation execution, MPI (GDDI) creates on each compute node just one rank. This rank usually does not have relevant computation; the actual computation is carried out by the team of threads that are spawned from this MPI rank using the OpenMP API. The role of the MPI rank is mainly to communicate with the other ranks in the other compute nodes (e.g., through send and receive protocols). Since threads in a team can efficiently share the node memory address, the MPI/OpenMP ansatz can minimize replicated data as well as the intranode communication overhead that subsequently enhances the computation efficiency and reduces the memory footprint. This approach has been applied to both HF and DFT codes by Mironov and co-workers.

In addition to the hybrid MPI/OpenMP model, the resolution-of-the-identity (RI) approximation[90–92] has been applied to correlated (fragmentation) methods, particularly to the second-order Moller–Plesset perturbation theory (MP2). The idea behind this combination is that the fragmentation methods chemically divide large molecules into "small" fragments; this is followed by the application of the RI approximation that further reduces the size of large data structures that arise from the underlying electronic structure calculations for fragments; finally, the hybrid parallel programming model minimizes replicated data and subsequently maximizes the available node shared memory. All of these factors maximize the locality of the computations by allowing the entire large data arrays or large chunks of them to be fit into the node memory. The data are then processed by thread workers enabled by the OpenMP API. The next paragraphs briefly discuss the MPI/OpenMP

implementation for regular and fragmentation HF and the RI-MP2 energy and gradient.

At the Hartree–Fock level of theory, the bottleneck of the calculations is the evaluation of four-index two-electron repulsion integrals (4-2ERI) in the AO basis. The AO basis functions on each atom that share certain common internal parameters (e.g., the angular momentum) are grouped into a shell. The integral evaluation, therefore, would need to loop over four shell (shell quartet) layers of AOs. Integrals of all AO basis functions in this shell combination are calculated at once. Before the shell quartet is executed, a fairly large number of small integrals can be eliminated[90,93] using the Cauchy–Schwarz inequality. Both symmetry and the screening can significantly reduce the computational cost of integral evaluation. After integrals in a shell quartet are calculated, they are accumulated into the Fock matrix.

In the original MPI-based HF code[94] in GAMESS,[1,2] all data arrays (e.g., overlap matrix, common blocks of AO shell information, Fock matrix, and density matrix) are replicated over all MPI ranks. A global sum is needed to accumulate the Fock matrix contribution from all MPI ranks at the end of the calculation. For the new generation of multiple core computers, the replicated arrays can introduce a very large memory footprint that deters the program from making use of all compute node resources efficiently.

There are two algorithms for the MPI/OpenMP HF implementation[32] in GAMESS, developed by Mironov and co-workers. The first approach is based on the private Fock matrix, and the second uses a shared Fock matrix for threads in a team. The private Fock matrix approach introduces better performance due to the direct accumulation of integrals to the Fock matrices, and it only needs one barrier at the end of the computation to reduce the private Fock matrix to the final one. For the shared Fock matrix approach, barriers are set up to prevent data race conditions (i.e., writing integrals to the same memory address of the shared Fock matrix). Apparently, the private Fock matrix method introduces a larger memory footprint than the shared Fock approach. Therefore, the shared Fock approach is useful when limited memory is a problem. Benchmark calculations for the MPI/OpenMP HF implementations for carbon-based material up to 2000 carbon atoms introduced a speed-up of ~6× compared with the original MPI-based code in GAMESS.

In the fragmentation context, particularly the FMO method,[26,36,95–98] the MPI/OpenMP HF implementation is directly helpful for FMO. For higher FMO orders, each fragment is submerged into the electrostatic potential (ESP) of the nuclei and the electron density of all other fragments. The most expensive part of the ESP is to evaluate the Coulomb interaction of electron densities among fragments. The ESP step has therefore been parallelized by Mironov.

For correlated fragmentation methods, e.g., the second-order Moller–Plesset perturbation theory method (MP2), one of the bottlenecks is the integral transformation from the AO to the MO basis, which is a matrix multiplication operation. While matrix multiplication is well supported by linear algebra libraries, the MP2 energy and gradient usually require large memory to store large data structures such as 4-2ERIs in the AO, the MO, and/or partially AO/MO bases. There are two main MP2 codes in GAMESS. The first (IMS) code[99] relies on storing partially and fully transformed integrals on disk files. The other (DDI) code[89,100] manipulates integral matrices on the distributed memory buffer. The DDI code is more efficient since the read/write from/to the distributed memory is more efficient than those on disk files. Another MP2 energy code[101] in GAMESS employs the resolution-of-the-identity (RI) approximation that approximates 4-2ERIs by the product of 3-2ERIs and 2-2ERIs. The computational cost of 3- and 2-2ERI integral evaluation is small (e.g., ~5%–10% of the total computational cost). For the MPI-based RI-MP2 implementation, when increasing the number of MPI processes, the data are usually split into smaller chunks for write/read operations and for subsequently feeding the matrix multiplication subroutine with smaller chunks of input data. Therefore, increasing the number of MPI ranks might implicitly reduce the overall performance.

Modern multicore compute nodes usually have ~64–72 cores with ~125 GB to 250 GB of memory/node. For the pure MPI model, if the number of MPI ranks created on each node is equal to the number of cores, each rank can only use ~1–2 GB in the memory address space. Additionally, each MPI rank needs copies of most data (e.g., common blocks for AO and auxiliary bases, MO vectors, density matrices). For calculations that need large memory, the pure MPI code has to be kicked off with a small number of ranks, which subsequently wastes a large number of CPU cycles. For the hybrid MPI/OpenMP model, only a small number of ranks (usually just one rank) are created on each compute node; each rank then spawns a team of threads that can share the same memory address space. Therefore, both memory and CPU cycles are used efficiently in the hybrid MPI/OpenMP model. This is particularly important in the context of the fragmentation methods since in most cases, large data structures in each fragment computation are usually well fitted to the node memory that completely removes the time-consuming write/read operation to/from disk file/distributed memory. Therefore, in most fragmentation RI-MP2 calculations, all fragments can be treated locally on one (logical) compute node that significantly improves the performance of the implementation. When data structures are not fit into the node memory, the large shared memory of the MPI/OpenMP model still facilitates large chunks of distributed arrays to be copied to node memory for computation, which is still much more efficient than copying small tiles of data many times from distributed arrays to the replicated arrays in a pure MPI treatment. Benchmark calculations on water clusters of ~2200 water molecules using 8–700 64-core KNL nodes showed that the new MPI/OpenMP FMO/RI-MP2 energy code[17] implemented in GAMESS has gained a speed-up of ~10×. For the gradient,[18] the speed-up is ~4–8×.

Since 2018, the US Department of Energy (DOE) has started operating and deploying GPU-based supercomputers with vendor optimized programming models such as CUDA, HIP, and SYCL. However, due to their limited functional portability, it is challenging for HPC application developers to maintain their applications in an efficient and effective way across various computer architectures. Directive-based programming models for accelerators can be a solution. In terms of the RI approximation, the computational core of the MP2 correlation energy evaluation is the matrix multiplication, which is supported by several GPU linear algebra libraries (e.g., NVIDIA cublas). The cost of 3-index and 2-index 2-electron repulsion integrals is about 5%–10% of the total cost. Therefore, in an initial effort to port GAMESS (Fortran) to GPUs,[102] all essential matrix multiplication operations in the RI-MP2 energy

kernel have been restructured and offloaded to GPUs using OpenMP and OpenACC GPU-offloading models and multiple linear algebra libraries. The benchmark calculations for clusters of 30–60 water molecules and fullerene (C60) show that the speed-up of the GPU RI-MP2 kernel on a single V100 GPU relative to the MPI/OpenMP RI-MP2 energy calculation on a P9 socket (22 cores, 88 threads) is ~20×; the speed-up relative to the pure MPI RI-MP2 energy code on a P9 socket (22 cores) is ~60×. This study has demonstrated that directive-based offloading implementations can perform near the GPU/CPU theoretical speed-up based on the machine peak ratios.

B. Coupled cluster methods

GAMESS allows for a wide variety of calculations based on the coupled-cluster (CC) theory and its extensions to excited, electron-attached, and ionized states via the equation-of-motion (EOM) formalism. This includes CC and EOMCC wave functions and energies as well as properties other than energy, and, in the ground-state case, larger polyatomic systems treated with the local correlation cluster-in-molecule (CIM) formalism.

1. Ground-state calculations

All of the GAMESS ground state CC options, which have been implemented in Refs. 103–111, are based on the exponential wave function ansatz[112,113] of the single-reference CC theory,[114–119] $|\Psi_0\rangle = e^T|\Phi\rangle$, where $T = \sum_{n=1}^{N} T_n$ is the cluster operator, T_n is the n-particle–n-hole (np–nh) or n-tuply excited component of T, N is the number of correlated electrons, and $|\Phi\rangle$ is the reference determinant defining the Fermi vacuum, which is usually obtained in HF calculations of the restricted (RHF), restricted open-shell (ROHF), or unrestricted (UHF) types.[120,121] The CC options in GAMESS allow for RHF[103–111] and ROHF[109–111] references, although the spin-integrated CC subroutines were written in a generic way, which could be interfaced with restricted as well as unrestricted references. The spin-adapted implementations of the closed-shell CC codes[103–108] are faster than the corresponding spin-integrated implementations by a factor of 2–3.

The ground state CC options in GAMESS include both the conventional approaches, such as the CC method with doubles (CCD) in full and linearized forms,[116–118,122,123] the CC approach with singles and doubles (CCSD),[124] where T is truncated at the $T_1 + T_2$ components, and the widely used perturbative CCSD(T) correction,[125] and the more robust renormalized CC (R-CC) and completely renormalized CC (CR-CC) triples corrections to CCSD.[103–109,126,127] The GAMESS CC options also include the conventional, renormalized, and completely renormalized CCSD(TQ) levels correcting the CCSD energies for a combined effect of the triply and quadruply excited clusters.[104–106,126–133]

Among the CR-CC methods, one that is especially important is the CR-CC(2,3) triples correction to CCSD,[107–109,134] which is at least as accurate as CCSD(T) for molecules near their equilibrium geometries and for non-covalent interactions, while being much more robust than CCSD(T) when chemical bonds are stretched or broken and when chemical reaction pathways are examined. CR-CC(2,3) is recommended as a substitute for CCSD(T), especially because computational costs of running CR-CC(2,3) are no more than twice the costs of the analogous CCSD(T) calculations. Another bonus of using CR-CC(2,3), as an alternative to CCSD(T), is the fact that, along with the accurate triples correction to CCSD, the user running CR-CC(2,3) gets access to the one-body reduced density matrix (1-RDM), right natural orbitals and their occupation numbers, Mulliken and Löwdin populations, bond orders, and electrostatic dipole moments, calculated at the CCSD level. The linearized and full CCD, CCSD(T), R-CCSD(T), CR-CCSD(T), R-CCSD(TQ), and CR-CCSD(TQ) are implemented in GAMESS for closed-shell RHF references only.[103–106] The CCSD and CR-CC(2,3) codes work for both RHF and ROHF reference determinants, allowing one to perform such calculations for closed- and open-shell systems,[107–109,134–136] obtain the CR-CCSD(TQ) and CR-CCSD(T) energies, and add the quadruples (+Q) correction, defined as [CR-CCSD(TQ)–CR-CCSD(T)], to the CR-CC(2,3) energy, as in the CR-CC(2,3)+Q approximation.[132,137]

One of the most recent additions to GAMESS, which is particularly helpful when the CR-CC(2,3) theory level is insufficient due to the more substantial coupling among the singly, doubly, and triply excited clusters, i.e., when the full CCSDT-type treatment is required but full CCSDT is too expensive, is the CC(t; 3) option.[110,111,138] In CC(t; 3), one corrects energies resulting from the active-space CCSDt calculations, in which T includes all singles (T_1), all doubles (T_2), and a subset of triples (a subset of T_3 amplitudes) defined using active orbitals,[139,140] for the remaining, predominantly dynamical, triple excitations that have not been captured by CCSDt. Having the leading T_3 amplitudes in it, the CCSDt approach alone is already often very accurate, especially when non-parallelity errors characterizing potential energy surfaces relative to its CCSDT parent are examined. CC(t; 3) improves the CCSDt calculations even further, being essentially as accurate as full CCSDT for both relative and total electronic energies, even in situations involving bond breaking, at a fraction of the computational cost.[110,111,138] CCSDt becomes CCSDT[141] when all orbitals used to select T_3 amplitudes are active. Therefore, the CCSDt codes in GAMESS allow one to run full CCSDT calculations as a byproduct. When the active orbital set (which the user defines in the input) is empty, CCSDt = CCSD and CC(t; 3) = CR-CC(2,3). The CCSDt and CC(t; 3) codes in GAMESS, which, unlike other GAMESS CC and EOMCC options, were implemented using automated formula derivation and implementation software, work for both RHF and ROHF reference determinants, allowing calculations for closed- and open-shell species.

2. Excited states

GAMESS can perform a variety of calculations for excited electronic states, which are based on the EOMCC wave function ansatz.[142] Among the EOMCC methods implemented in GAMESS are the basic EOMCCSD approximation,[142] available for both RHF and ROHF references,[143–147] and the variety of CR-EOMCC and δ-CR-EOMCC triples corrections to the EOMCCSD total and excitation energies,[127,143–147] which can be run at this point for RHF reference only. If the user is interested in non-singlet states of a closed-shell system or singlet as well as non-singlet states obtained in a single calculation, using the open-shell EOMCCSD codes with the ROHF $S = 0$ reference determinant is the only option in GAMESS at this time.[147] One can also use the open-shell EOMCCSD/ROHF codes for excited states of molecules with non-singlet (e.g., doublet)

ground states, but one has to keep in mind that the resulting wave functions will not be spin adapted. If the user is interested in rigorously spin-adapted CC/EOMCC calculations for the ground and excited states of radicals or systems that can formally be obtained by adding one electron to or removing one electron from the corresponding closed-shell core, choosing the electron-attachment (EA) and ionization potential (IP) EOMCC options is the best idea.

EOMCCSD is reasonably accurate for excited states dominated by one-electron transitions, but it fails whenever the excited states of interest have significant double excitation character or excited-state potentials along bond breaking coordinates are examined, producing errors in the excitation energies that usually exceed 1 eV, being frequently much larger.[127,129,131,143–145,147,148] Even when excited state wave functions are dominated by one-electron transitions, EOMCCSD is not fully quantitative, giving errors on the order of 0.3–0.5 eV in many cases.[148] One can rectify these problems by turning to higher EOMCC levels, represented in GAMESS by the aforementioned CR-EOMCC and δ-CR-EOMCC triples corrections, which are more robust, especially when two-electron excitation components become more substantial, than the perturbative methods of the EOMCCSD(T)[149,150] or CC3[151] type.[127,129,131,143,147]

3. Electron-attached and ionization-potential equation-of-motion coupled-cluster approaches

One of the most useful features of the EOMCC wave function ansatz is the possibility to extend it to open-shell systems around closed shells, such as radicals and cations or anions of closed-shell species, which can formally be obtained by attaching an electron to or removing an electron from the underlying closed-shell core. This can be done by replacing the particle-conserving form of the R_μ operator of EOMCC, which excites electrons from the occupied to unoccupied orbitals in the reference, by its particle-nonconserving EA (electron attachment) or IP (ionization) extensions. Due to the use of a closed-shell reference wave function, which in the EA and IP EOMCC GAMESS options,[152–154] is the CCSD ground state of the underlying closed-shell core, the EA-EOMCC and IP-EOMCC methods provide an ideal framework for performing orthogonally spin-adapted calculations for radicals and cations or anions of closed-shell species. They are especially useful in determining the electronic spectra of radicals[145,152–154] and photoelectron spectra.[155,156]

The EA and IP EOMCC options in GAMESS include EA-EOMCC(2p–1h) and EA-EOMCC(3p–2h) in the EA case and IP-EOMCC(2h–1p) and IP-EOMCC(3h–2p) in the IP case, where symbols in parentheses indicate the truncation level. The higher-level EA-EOMCC(3p–2h) and IP-EOMCC(3h–2p) approaches are especially useful, since they prevent failures of the basic EA-EOMCC(2p–1h) and IP-EOMCC(2h–1p) approximations when the relevant electron attachment/ionization processes are accompanied by significant electron relaxation effects in the closed-shell core, which is the case in nearly all electronic states of radicals[152–154] and in the electron attachment and ionization processes in photoelectron spectroscopy involving higher-energy shake-up states.[155,156] When running the higher-level EA-EOMCC(3p–2h) and IP-EOMCC(3h–2p) calculations, computational costs may become a significant bottleneck. In the spirit of other active-space EOMCC methods,[140,157–161] this issue is addressed in GAMESS by using active orbitals to select the dominant $R_{\mu,3p\text{-}2h}$ and $R_{\mu,3h\text{-}2p}$ components.[152–154] It is recommended to use the active-space EA-EOMCC(3p–2h) and IP-EOMCC(3h–2p) approaches, which have costs on the order of CCSD or EOMCCSD times a small prefactor.

4. Properties other than energy

GAMESS CCSD and EOMCCSD codes allow for analytic calculations of properties other than the energy[142,144] (available for closed-shell systems, as described by RHF orbitals). GAMESS prints a number of useful ground and excited state properties, such as dipole moments, Mulliken and Löwdin populations, bond orders, natural orbitals and natural orbital occupation numbers, transition dipole moments, and dipole and oscillator strengths, to name a few examples. Since the CC and EOMCC 1-RDMs are not Hermitian, calculations of the dipole and oscillator strengths require that the relevant $|\langle\Psi_\mu|\theta|\Psi_\nu\rangle|^2$-type expressions are represented as $\langle\Psi_\mu|\theta|\Psi_\nu\rangle\langle\Psi_\nu|\theta|\Psi_\mu\rangle$, which is exactly what GAMESS does. In analogy to the CC and EOMCC states, one has to distinguish between the left and right natural orbitals in determining, for example, many-electron densities. To minimize the amount of output, only right natural orbitals are printed in the main output file. This is not a major limitation though, since GAMESS also prints the complete set of 1-RDMs and transition 1-RDMs $\gamma^q_p(\mu,\nu)$, as defined above, in a RHF molecular orbital (MO) basis in the auxiliary output file. Electrostatic properties, such as dipole moments and (hyper)polarizabilities, can also be determined using finite-field calculations. Geometry optimizations and transition-state searches can be performed using numerical derivatives.

5. Local correlation cluster-in-molecule approaches

The CC and EOMCC calculations using canonical RHF, ROHF, or other delocalized MOs may become prohibitively expensive when larger many-electron systems are considered. For example, most of the methods described above have computational steps that scale as the sixth or seventh power of the system size, N, with memory requirements scaling as N^4. This is addressed in GAMESS with the help of fragmentation methods, as discussed in Sec. II, and the local correlation CIM methodology,[23,162–168] which is capable of reducing the high polynomial costs of CC calculations using delocalized HF orbitals to steps that scale linearly (or even sublinearly) with the system size, N.

The basic idea of all CIM-CC and CIM-MPn methods,[23,162–168] including those implemented in GAMESS,[23,163–167] is the observation that the total correlation energy of a large system or any of its components, such as the triples correction of CCSD(T) or CR-CC(2,3), can be obtained as a sum of contributions from the occupied orthonormal localized MOs (LMOs) and their respective occupied and unoccupied orbital domains that define the CIM subsystems.

All CIM approaches result in straightforward algorithms in which, beginning with the AO → MO integral transformation and ending up with the final CC or MPn work, the CC or MPn calculation for a large system is split into independent and relatively inexpensive calculations, in analogy with other fragmentation approaches for CIM orbital subsystems, which can easily be executed in parallel (on multiple cores or multiple nodes, or both). The final correlation energy of the entire system is determined by adding

correlation energy contributions extracted from the calculations for the individual CIM subsystems. They are characterized by the linear scaling of the computational time with the system size, when a single-level CIM-CC or CIM-MP2 approach is used,[23,163,164,166] memory requirements that do not grow with the size of the system,[23,164,166] coarse-grain parallelism, which can be further enhanced by the fine-grain parallelism of each CIM subsystem calculation, and the purely non-iterative character of the local triples and other perturbative energy corrections, which is achieved in GAMESS via the concept of quasi-canonical subsystem MOs.[23,164] They can be made even less expensive, leading, *de facto*, to sublinear scaling algorithms, when multi-level CIM schemes mixing higher- and lower-order methods are employed.[165] The CIM methodology implemented in GAMESS also allows one to combine canonical (e.g., MP2 or CCSD) calculations for the entire system, which can be run in parallel, with local calculations for subsystems that require a higher-level [e.g., CR-CC(2,3)] correlation treatment.[167]

The CIM methods implemented in GAMESS include MP2, CCD, CCSD, CCSD(T), and CR-CC(2,3) for closed-shell systems and CCSD and CR-CC(2,3) for open shells. The main parameter ζ controlling the design of CIM subsystem domains can be varied by the user (the canonical limit is obtained when $\zeta \to 0$), although GAMESS provides a default, which is often a good starting point. The GAMESS CIM codes can be executed sequentially or in parallel, and they can be combined with the FMO method, as described in Sec. II.

C. ORMAS, ORMAS+MP2, CEPA

The Occupation Restricted Multiple Active Space (ORMAS)[22,169] approach is a configuration interaction (CI) method that, as the name implies, (1) divides the orthogonal orbitals of a system into a number of ORMAS groups (OGs) and (2) allows the electron occupation of each OG to vary between minimum and maximum limits. The number of OGs, their constituent orbitals, and occupation minima/maxima can be arbitrarily chosen by the user (within logical limits). In this way, a very diverse set of CI and MCSCF wave functions can be constructed and optimized. The implementation is determinant based, direct, and parallel so that several billion determinants can be included in a calculation. The types of wave functions that can be optimized include ORMAS0 (constant number of electrons in each OG, e.g., groups of bonding/antibonding orbitals), ORMAS0-SD (SD = single and double excitations out of the ORMAS0 space), and CIx/MR-CIx (x = desired maximum electron excitation level) for which orbital optimization is also possible. More recently, single reference (SR) and multireference (MR) coupled electron pair approximation (CEPA) methodologies were added to the ORMAS module.[170] Three popular approaches are available: CEPA(0),[171] average coupled pair functional (ACPF),[171] and averaged quadratic coupled-cluster (AQCC).[172]

A significant enhancement to the ORMAS method is the ability to include second-order perturbation theory energy corrections (ORMAS-PT2).[24] Then, large active spaces can be used in MCSCF reference functions, e.g., full valence or full π, and dynamic correlation subsequently accounted for via PT2 corrections in the style of MRMP2/MCQDPT2.[173,8] Thus, for large systems it is possible to cheaply compute accurate properties such as binding/dissociation energies, transition state barrier heights (including for bond forming/breaking), and excited state energies. With regard to the latter, one efficient route is to use state-averaged MR-CISD reference wave functions (full-valence or $-\pi$) in which the occupied orbitals are optimal. Additionally, energies of different spin states can be simultaneously determined. Another useful feature is that solvent effects can be included via the PCM[4] through MCSCF wave function optimization[174] and one-electron integral modification. The ORMAS-PT2 implementation follows the style of the analogously programmed MRMP2/MCQDPT methods[175] and is determinant based, direct, and parallel.

D. Correlation energy extrapolation by intrinsic scaling

The Correlation Energy Extrapolation by Intrinsic Scaling (CEEIS) method of Bytautas and Ruedenberg[176] is a powerful procedure for the recovery of the full configuration interaction energy, E^{FCI}. CEEIS is based upon the exact expansion of the FCI energy as a sum of CIx excitation level energy contributions,

$$E^{FCI} = E(0) + \Delta E(1,2) + \sum_{x \geq 3} \Delta E(x), \tag{5}$$

where E(0) is the reference energy (single determinant or multiconfigurational) and excitation levels are shown in parentheses. The energy difference $\Delta E(1,2) = E(2) - E(0)$ represents the CISD (or CI2) correlation energy, and $\Delta E(x)$ ($x \geq 3$) denotes the energy lowering when going from CI($x-1$) to CIx, i.e., $\Delta E(x) = E(x) - E(x-1)$. In the CEEIS method, $\Delta E(1,2)$ and $\Delta E(3)$ are computed exactly and $\Delta E(x)$ for $x > 3$ are extrapolated from energy differences $\Delta E(x|m) = E(x|m) - E(x-2|m)$. The latter two quantities are obtained from CIx and CI($x-2$) computations in which electrons are only allowed to excite into a number of active virtual orbitals m that is less than the total number M. The crux of the CEEIS method is the discovery that $\Delta E(x-2|m)$ and $\Delta E(x|m)$ are linearly related as m approaches M, so that $\Delta E(x) = \Delta E(x|M)$ can be determined from $\Delta E(x-2)$, and $\Delta E(x-2|m)$, $\Delta E(x|m)$ over a range of m. Therefore, when it is not possible to compute CIx energies in the full basis, they can be accurately determined via far cheaper computations. Furthermore, by gradually increasing x, estimates of FCI energies can be obtained.

An important consideration for CEEIS is the generation of appropriate virtual orbitals following optimization of the reference wave function. The recommended approach is to perform a preliminary CISD calculation, compute the corresponding one-particle density matrix, and diagonalize the virtual–virtual block to obtain natural orbitals for the virtual space (VSDNOs). These VSDNOs are then ordered according to decreasing occupation numbers.

An automated CEEIS procedure has been implemented in GAMESS where single determinant and MCSCF zeroth-order functions can be used. The values of x and m are specified by the user; however, these should be chosen carefully so that the changes in $\Delta E(x-2|m)$ and $\Delta E(x|m)$ are linearly proportional. Ideally, the full CISDT (or CI3) energy should be computed for high accuracy, but if this is not possible, it can be extrapolated from the CISD energy. The CEEIS method has been used to determine benchmark-quality ground state properties for a variety of molecules[176–180] and has also been generalized for multiple electronic states via the use of state

averaged reference functions.[181,182] CEEIS has also been utilized for the identification of compact and accurate CI wave functions.[8]

E. Analysis of complex wave functions by reconstruction in terms of quasi-atomic orbitals

While accurate computations of energetics and properties are an essential goal of *ab initio* methods, equally vital is the deduction of insights in order to (1) translate the complex wave functions into elementary, familiar bonding concepts, and (2) conceptualize rules and trends in chemistry. Over the last several decades, Ruedenberg and co-workers have evolved comprehensive approaches to reconstitute molecular wave functions and energies in terms of quasi-atomic orbitals. Many of these methods have been incorporated in the GAMESS package, and a synopsis of the available tools follows.

In the late 1970s, Ruedenberg *et al.* showed that Full Optimized Reaction Space (FORS) wave functions (i.e., full-valence active space MCSCF) intrinsically incorporate a set of minimal basis orbitals that resemble deformed, quasi-atomic orbitals (QUAOs).[183–185] Subsequent to FORS optimization, QUAOs can be generated in several ways including (1) direct localization of the molecular orbitals (MOs), for which there are several available approaches in GAMESS including the Edmiston–Ruedenberg method,[186] and (2) optimal alignment of the MOs to free-atom orbitals via singular value decomposition (SVD).[12] QUAOs can also be formulated for wave functions that are simpler than FORS, viz., Hartree–Fock[12] and less than full valence MCSCF,[187] and rely on the generation of valence-virtual orbitals (VVOs).[80,188] The VVOs are extracted from the unoccupied virtual orbitals so that they, together with the occupied orbitals, span an orbital space that is an excellent approximation to the full valence, or internal, space. Once obtained, it is usually necessary to *orient* the QUAOs on each atom so that they exhibit the global bonding pattern of the molecule (e.g., form bonds with other atom QUAOs or become lone pairs).[189,190] This orientation is accomplished with a completely unbiased, purely mathematical, method that uses no intuitive information about the molecule whatsoever. Finally, the first-order density matrix is expressed in terms of the oriented QUAOs to reveal qualitative and quantitative chemical data such as atom charges, non-bonding/inactive orbitals, and bond types and strengths. Covalent bond strengths can also be quantified by a new measure called the kinetic bond order that calculates the energy lowering due to interference between oriented QUAOs.[187]

All of the aforementioned methods are available in GAMESS and have been used to study a series of diverse molecules to elucidate the inherent bonding patterns at minima and along reaction surfaces.[191–196] More recently, the methodology has been expanded to sixth row atoms.[197,198] Complex techniques that resolve binding energies into intra-atomic and interatomic parts have also been formulated and utilized to uncover the physical origins of covalent binding.[199,200]

F. Spin-flip and spin-correct spin-flip

In most cases, the proper description of non-dynamic correlation requires the use of multi-reference methods.[201] Although several multi-reference methods are available in GAMESS, their exponential cost makes them computationally prohibitive, limiting such methods to relatively small systems and small active spaces. The spin-flip (SF) family of methods, introduced by Krylov in 2001,[202–204] was developed as a possible alternative to multi-reference methods, without the multi-reference cost. In contrast to conventional multi-determinant approaches, SF methods rely on a high-spin reference determinant ($M_S > 0$), which, through a series of spin-flipping excitations ($\Delta M_S < 0$), generates a multi-determinant wave function of a lower multiplicity. The multi-determinant nature of the final wave function, as well as the high-spin starting orbitals, allows SF methods to capture multi-reference effects within a single-reference formalism. Spin-flip has been implemented within several quantum chemistry methods, including configuration interaction (SF-CI),[202,205,206] time-dependent density functional theory (SF-TDDFT),[207] and coupled cluster (SF-CC).[203,208,209]

Figure 1 gives a graphical representation of the single spin-flip procedure.

Due to their simplicity and speed, SF-CIS and SF-TDDFT (implemented within the Tamm–Dancoff approximation) are the most popular iterations of the SF methods. Both methods are available in GAMESS. This includes energies and analytic gradients, as well as solvent effects through PCM,[210] or the effective fragment potential (EFP).[59,211] These methods have been used to successfully describe bond-breaking, transition state geometries, excited states, and geometries of conical intersections, both in the gas phase[9,212] and in solution.[213]

A significant disadvantage of SF methods is that they suffer from spin-contamination. The spin-flip procedure shown in Fig. 1 ensures that the final SF wave function is an eigenfunction of the $\hat{S}_Z$ operator, but not necessarily an eigenfunction of the $\hat{S}^2$ operator. This is evident from the second half of determinants (v–viii) in Fig. 1. In consequence, the final SF wave function is often a mixture of different multiplicities. Moreover, the spin-contamination is inconsistent and often hard to predict, particularly at geometries where degenerate configurations are important. Because of its drawback, a variety of approaches have been suggested to correct the spin-contamination of SF methods.[205,206,214,215]

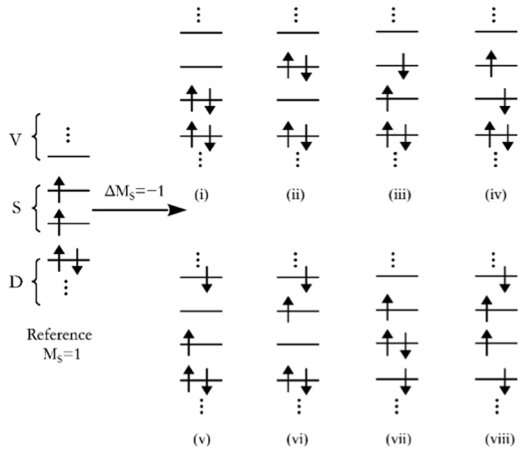

FIG. 1. A visual diagram of a single spin-flip procedure.

The SF-ORMAS method[11,216] was introduced in GAMESS to correct the spin-contamination problem inherent in SF-CI methods. As the name suggests, SF-ORMAS is the spin-flip variant of the ORMAS-CI method, introduced by Ivanic in 2003.[22,169] ORMAS is a general determinant-based CI algorithm that allows for the partition of the orbital space into arbitrary subspaces, each constrained by a minimum and maximum electron occupation. This makes a variety of CI schemes possible within a single computational formalism. The SF-ORMAS variant functions similarly to the ORMAS-CI method but imposes the additional constraint that all generated determinants must be of a lower multiplicity ($\Delta M_S < 0$) than that of the reference determinant (i.e., the "spin-flip" constraint).

The SF-ORMAS method not only corrects the spin-contamination problem, but due to the flexibility of the ORMAS algorithm, allows for a variety of SF-CI schemes. SF-ORMAS can be supplemented with a perturbation correction (termed SF-MRMP2) to account for dynamic correlation that is normally neglected from most SF schemes. Energies are available in both the gas phase and in solution (via the EFP or PCM methods), whereas analytic gradients are available only for the gas phase.[216] Recently, non-adiabatic coupling matrix elements (NACMEs) were also implemented for the SF-ORMAS method.[217]

SF-ORMAS was shown to successfully describe minimum and transition state geometries, diradical states, single and multiple bond-breaking, and low-lying excited states, with accuracies often matching those of methods such as CASPT2 and MRCI.[11] Conical intersections optimized with SF-ORMAS are comparable to those optimized by multi-reference methods. The recently implemented NACME also shows good qualitative accuracy compared to the NACME of methods such as CASSCF and MRCI.[217] This strongly suggests that the SF-ORMAS method is suitable for the study of non-adiabatic effects.

G. Quantum Monte Carlo

As part of a Department of Energy Exascale Computing Project (ECP), the GAMESS EFMO code has been interfaced with the quantum Monte Carlo (QMC) program suite QMCPACK. This combined QMC-EFMO method[218] inherits the advantages of the two methods: the high accuracy of QMC and favorable computational scaling of EFMO.

The QMC method is a family of stochastic approaches for solving the Schrodinger equation.[219] The statistical uncertainties of the predicted QMC properties (e.g., the energy of a molecule) can be estimated and controlled. Thus, the QMC results are typically very reliable, with an accuracy that is typically below 1 kcal/mol.[220,221]

The QMC method has a favorable scaling of computational time with respect to the number of electrons that is close to cubic.[222,223] In addition, the QMC algorithms due to their stochastic nature are easy to make parallel and are consequently ideally suited for massively parallel computers. The QMC parallelization compensates for the fact that the pre-factor, i.e., the constant of proportionality in front of the cubic scaling factor, is significantly larger than that for HF and DFT. Overall, due to its high accuracy and favorable computational scaling, the QMC method is an attractive alternative to the more traditional *ab initio* methods.

The computational scaling of QMC can be substantially reduced by the use of fragmentation methods. In the QMC-EFMO method, the energy is computed as in any EFMO computation but with QMC instead of a post-HF method for the correlation energy of the monomers and dimers.

The QMC-EFMO method is implemented through an integration of GAMESS and QMCPACK programs. The molecular system is first fragmented in GAMESS. Next, a stream of QMC correlation energy evaluations on monomers and dimers are done in parallel by an initial calculation by GAMESS followed by a sequence of QMCPACK calculations. Ultimately, all calculations are assembled in the final QMC-EFMO energy result.

The QMCPACK–GAMESS integration is based on Python and Fireworks.[224] Fireworks is a workflow automation package written in Python that utilizes the MongoDB database system.[225] The Python/Fireworks scripting automates the workflow of GAMESS and QMCPACK input files creation, program runs, and output files parsing and thus achieves a seamless integration of the two programs.

A double-basis approach following Ref. 226 is used in the QMC-EFMO calculations. For the QMC correlation energy, i.e., the QMC calculations by QMCPACK and the preliminary HF by GAMESS, the Burkatzk–Filippi–Dolg (BFD) effective core potential basis set[227] is used. For the generation of the EFP parameters in the initial EFMO calculation by GAMESS, the 6-311++G(2df,2pd) basis set is used.

QMC-EFMO is a method with computational scaling that is close to linear while retaining almost entirely the QMC correlation energy. The QMC-EFMO method is illustrated on ground-state calculations on a four-water cluster, a set of larger water clusters, and the excitation energy of micro-solvated acetone. In all of these examples, QMC-EFMO reproduces the full QMC correlation energies and excitation energies very well.

H. Density functionals

GAMESS provides access to many popular density functional approximations across the five rungs of Jacob's ladder[228] [e.g., local density approximation (LDA), generalized-gradient approximation (GGA), meta-GGA, hybrid GGA/meta-GGA, and double hybrid]. The 2012 excited-state benchmark by Leang *et al.*[229] demonstrated several density functional approximations available in GAMESS, which have been implemented for both ground- and excited-state calculations. Several new density functional approximations have been added to GAMESS since the study by Leang *et al.*, most notably several variants of the Minnesota meta-GGA density functionals: M11,[230] M11-L,[230] MN12-L,[231] MN12-SX,[232] MN15,[233] and MN15-L.[234] In addition to several revised versions of the Minnesota meta-GGA functionals: revM06,[235] revM06-L,[236] and revM11.[237] Of the Minnesota family of density functional approximations available in GAMESS, only M11, M11-L, revM06, revM06-L, and revM11 are limited to ground state calculations.

I. DFTB

Fragmentation methods greatly reduce the computer time requirements for high-level *ab initio* and first principles energies and gradients and allow their computation for large-scale systems, thanks to near-linear scaling behavior with system size and efficient parallelization techniques. Long time scale molecular dynamics (MD) simulations on the other hand are still difficult to perform

even with the help of systematic fragmentation approaches.[238] This is because even the smallest possible fragment calculation of analytical gradients requires typically minutes, even on the fastest supercomputers available today, which means that a nanosecond MD trajectory with million integration time steps would have to run for almost one calendar year. Time cannot easily be "parallelized," in particular, when systems are studied in nonequilibrium or when the time scales of chemical processes, such as diffusion, are inherently slow. Therefore, it is necessary to use a computationally less expensive electronic structure method that reduces the time for the calculation of fragment energies and gradients by at least one order of magnitude.

One such method that has recently become very popular is the density-functional tight-binding (DFTB) method,[239] since it avoids the expensive calculation of electronic integrals by the way of a two-center approximation, the use of a minimal valence electron basis set, and Hamiltonian and overlap matrix elements, as well as diatomic repulsive potentials. These parameters are all tabulated as a function of interatomic distances for each chemical element combination. The DFTB method comes in a range of flavors, characterized by the order to which the electronic charge density fluctuation is expanded in a Taylor series around a reference density (typically a superposition of atomic densities). DFTB1 is accurate to first order and does not depend on charge densities;[240] DFTB2 is accurate to second order and contains Coulomb interactions between charge fluctuations;[241] and DFTB3 is accurate to third order and contains additionally a charge-dependent on-site self-interaction and a modification of the second-order Coulomb interaction term.[242] In addition, spin-polarization can be introduced in all three flavors via the introduction of an interaction term between spin populations in different atomic orbital shells, resulting in SDFTBn methodologies.[243] Finally, a range-separated treatment of the exchange energy was recently introduced as the LC-DFTB2 flavor in analogy to, for instance, LC-ωPBE, where the long-range correction "switches on" the Hartree–Fock exchange.[244] Fully analytic second-order energy derivatives are available for all DFTB versions except for LC-DFTB2, providing rapid and robust simulation of infrared and Raman spectra even for open-shell systems.[245] An implementation of time-dependent DFTB (TD-DFTB) for the computation of the UV/Vis absorption and emission spectra of systems containing several hundred atoms is also available along with analytic first-order energy derivatives, with and without the addition of the polarizable continuum model (PCM).[246]

The computational bottleneck of DFTB is associated with finding a self-consistent solution to the charge (and spin) density fluctuations, which requires solving the generalized Kohn–Sham eigenvalue equations in the tight binding framework. This step scales cubically with system size, similar to the parent DFT method, and hence fragmentation is ideally suited to reduce this unfavorable scaling. The resulting FMO-DFTB methods have been implemented in GAMESS[247–249] and allow quadratic time-to-solution for the calculation of quantum chemical atomic forces for very large systems.[250] Both two- and three-body FMO expansions are available for all DFTB versions.[251] The code allows the use of the velocity Verlet time integration algorithm for the long time scale MD simulations of complex systems, such as, for instance, peptide folding dynamics,[248] and a replica-exchange MD approach was also recently implemented within GAMESS for use with DFTB to allow for more efficient phase space sampling.[35] This makes the DFTB-based quantum chemical computation of free energy changes as a function of some inter- or intramolecular coordinate [potential of mean force (PMF)] possible at the selected level of DFTB or FMO-DFTB.

It is often said that "there is no free lunch," and this is certainly the case for FMO-DFTB as well. The electronic and repulsive potentials have to be optimized for the required chemical element combinations and the desired DFTB version, and the parameter optimization toolkit[252] will be released soon. In recent years, machine learning parameterization techniques have been developed and employed that improve the performance of the DFTB flavors such that results comparable to traditional density functional theory (DFT), correlated electronic structure methods, or experimental data can be obtained.[252] However, no matter to what degree the DFTB parameters are optimized, the requirement of parameter transferability will always result in systematic errors originating from the DFTB approximations themselves, such as the use of a minimal basis set or the two-center approximation. To mitigate this remaining systematic bias, Δ-machine learning methodologies[253] based on Behler–Parrinello neural network (NN) corrections for DFTB energies and forces have been developed.[254,255] Since systematic bias is less dependent on a given chemical system or geometric configuration, the DFTB+ΔNN approach is able to extrapolate from rather than interpolate among training data. MD simulations based on FMO-DFTB+ΔNN are, therefore, expected to achieve first principles or even higher-level accuracy for the predictive study of the dynamics of chemically complex systems.

J. Parallel coupled cluster

Coupled cluster (CC) theory[121] provides very accurate results in the computation of molecular energies and properties. The CC method truncated at the single and double excitation level (CCSD) and augmented with a noniterative perturbative treatment of triple excitations, viz., the CCSD(T) method,[125,256] is an accurate method in quantum chemistry. Unfortunately, the steep scaling of the computational costs of the CC methods, e.g., N^6 for CCSD and N^7 for CCSD(T), where N is a measure of the system size, restricts their applicability to chemically relevant problems. Adapting CC implementations to modern parallel computing architectures can effectively surmount this barrier. The primary goal of a parallel CC algorithm[13,16,257–259] is to make an efficient utilization of the total aggregate memory of a parallel computer for storing memory demanding quantities, thus affording computations involving large molecules and basis sets.

The existing parallel CCSD(T) implementation[13] in GAMESS is based on the third generation of the Distributed Data Interface[89] (DDI/3), which introduced shared memory capabilities for multiprocessor nodes on top of the multinode distributed memory model. The parallel CCSD(T) algorithm uses three types of storages for the requisite quantities: (a) distributed storage for large two-dimensional arrays over a number of nodes in a parallel computer (distributed memory), which has the largest storage capability and also bears the largest communication overhead, (b) the shared memory of each multiprocessor node, which can be directly accessed and modified by all intranode processes, and (c) replicated memory

of the parallel processes on a node, which has the smallest storage capacity.

In the current GAMESS algorithm, the various classes of two-electron repulsion integrals (2-ERIs) involving up to three virtual molecular orbital (MO) indices are stored in the distributed memory. The four-virtual integrals ([VV|VV]), which present a memory bottleneck, are not stored at all. The terms in the CCSD amplitude equations involving these 2-ERIs are rather computed via an atomic orbital (AO) integral-direct algorithm. Arrays of size scaling as N^2 and N^3, e.g., the T_1 amplitude matrix, are stored in the replicated memory of each parallel process. On the other hand, the T_2 amplitude matrix (storage scaling as $N_o^2 N_v^2$, where N_o and N_v denote the number of occupied and virtual MOs, respectively) is stored once per node in its shared memory address. The workload pertaining to the evaluation of the terms in the factorized CCSD amplitude equations is distributed over nodes according to the distributed storage of the 2-ERIs. The workload on each node is further distributed among the intranode processes. Importantly, the DDI/3 model employs Unix System V semaphores for intranode communication rather than a thread-based model (e.g., the OpenMP API[260]).

While the current parallel CCSD(T) algorithm was demonstrated to achieve reasonable scalability for chemically interesting problems in the limit of a large number of compute nodes, there is room for further improvements. Current efforts are focused in this direction. The distributed storage for the three virtual-one occupied integrals ([VV|VO]) in the current parallel CCSD(T) algorithm presents a memory as well as communication bottleneck. A pragmatic approach to reduce this bottleneck is to implement an AO integral-direct algorithm[261] for the terms that involve these 2-ERIs. Unlike the [VV|VV] integrals, the [VV|VO] integrals appear in a larger number of terms in the CCSD amplitude equations. A judicious regrouping of the various terms is thus important for an efficient evaluation; for example, to compute one group of terms involving the [VV|VO] integrals *simultaneously* with the evaluation of terms involving the [VV|VV] 2-ERIs. The existing code segments will be retained as much as possible such that the AO integrals need not be evaluated repeatedly.

Following the lead of the new RI-MP2 code discussed in Sec. III A, the parallel CCSD(T) code will make use of a hybrid DDI/OpenMP model by substituting the process-based parallelism on each node with thread-based parallelism. In the DDI/3 model, collective synchronizations over all intranode processes are applied in order to retain the integrity of the data stored in the shared memory address of the node. With an increasing number of intranode processes, the increased synchronization overhead becomes competitive with the enhanced distribution of the computational workload per node. For this reason, the intranode scalability of the existing parallel CCSD(T) algorithm was found to be less than optimal.[13] The synchronization overhead can be reduced by limiting the number of intranode processes to only a few (ideally one). Each process then gets a larger amount of local memory, which permits a larger amount of data to be replicated among them. The workload on each process can then be suitably parallelized via OpenMP threads. As threads communicate through a shared memory pool, an efficient parallelization at a significantly lower interprocess communication cost can be achieved by assigning a large team of threads to each process.

Further improvements in the parallel CCSD(T) implementation can be achieved by making use of graphical processing units (GPUs) to perform certain computations, which are both time consuming and memory expensive. GPUs facilitate a massive parallelization of logically simple computational steps at very high speeds. Contractions involving the [VV|VV] and [VV|VO] integrals with cluster amplitudes will be performed by offloading these computations to GPUs. This will require enabling GPU offloading capabilities within the parallel CCSD(T) algorithm. The use of modern OpenMP standards will make this feasible. An alternative strategy would be to obtain the 2-ERIs from the GPU-enabled integral library named LibAccInt, which is currently under development. This will accelerate the integral evaluation step. Furthermore, all steps involving contractions of the 2-ERIS with cluster amplitudes could be offloaded to GPUs for the maximum speed-up.

Efficient parallel algorithms will also be developed for the existing sequential CR-CC(2,3) implementation in GAMESS. As noted above, the CR-CC(2,3) approach[107] includes a noniterative correction for triple excitations on top of the CCSD energy via the method-of-moments ansatz. For developing a parallel CR-CC(2,3) algorithm, the key step is to parallelize the triples correction part. A hybrid DDI/OpenMP model will be used for this purpose. Further current developments in the CC methods within GAMESS include the implementation of analytic gradients[262,263] for the CCSD(T) and CR-CC(2,3) methods. Massively parallel algorithms will be developed for gradient calculations using similar parallelization models as outlined above.

Another important current development in the CC methodologies within GAMESS concerns a massively parallel implementation of the CCSD(T) and CR-CC(2,3) methods employing the resolution-of-the-identity (RI) approximation[90,264] for the 2-ERIs. Within the RI approximation, the 2-ERI matrix is approximated as products of three-index tensors. The storage requirements for the three-index integrals scale as $N^2 N_{aux}$, with N_{aux} denoting the size of the auxiliary basis set, in contrast to the N^4 storage requirements for the conventional four-index 2-ERIs.

The straightforward way to implement RI-CC methods[265–269] would be to assemble and store the four-index 2-ERIs prior to the iterative solution of the CCSD amplitude equations. While this would allow for the use of the existing CC implementation, such an algorithm does not take advantage of the reduced storage of the 2-ERIs. An alternative strategy is to assemble the four-index 2-ERIs as they are needed. Such an integral-direct algorithm bypasses the large storage requirements for the 2-ERIs. However, the repeated integral assembling steps in every iteration, the computational cost of which scales as N^5, should be minimized for an optimum efficiency. This can be achieved by regrouping the terms in the CCSD amplitude equations and formulating them in terms of intermediates, which involve contractions between the three-index 2-ERIs and the cluster amplitudes.[266,269] The use of these intermediates enables avoiding a direct evaluation of the terms involving the [VV|VO] integrals. The evaluation of the term involving the [VV|VV] integrals still remains the rate-determining step in the RI-CCSD calculation. An AO integral-direct algorithm will be developed for this purpose in which the four-index AO integrals will be assembled from the prestored three-index AO integrals.

With the above strategy to fully exploit the reduced storage requirements for the 2-ERIs, a parallel implementation of the RI-CC methods would require significantly less distributed data storage compared to the existing parallel CCSD(T) implementation. The use of the total aggregate memory of a parallel computer for storing the 2-ERIs, as exploited in the DDI/3 model, is less important for the RI-CC methods.[19,97,98,270] The GDDI model partitions compute nodes into groups, the size of which can be assigned according to the needs at runtime. All quantities required for the RI-CC calculations will be replicated among the groups, in this way eliminating the intergroup communication overhead. The three-index 2-ERIs will be distributed within each group. The workload on each node will be distributed over a small number of processes so as to maximize the associated shared memory per process. The computation on each process will then be parallelized using teams of OpenMP threads. One important advantage of using this hybrid GDDI/OpenMP model is the scope of combining the RI-CC methods directly with the FMO approach, which is the final goal of this work.

K. Interoperability

There are a few robust, no-cost or open-source, electronic structure packages available for *ab initio* molecular electronic structure computations. Among those program suites, GAMESS,[1] NWChem,[271] PSI4,[272] and CFOUR[273] stand out as flagship development platforms to perform highly accurate quantum chemical computations and implement new electronic structure approaches/models. Each of these programs contains millions of lines of computer codes and has unique functionalities and capabilities that have been developed over many years through the efforts of many researchers. For example, GAMESS has been evolving for almost four decades. However, there are tasks in *ab initio* electronic structure computations/models that are common to all program suites. In order to minimize further development efforts (minimize duplicate efforts) and to maximize the efficacy of the unique features, GAMESS has been interfaced with quantum chemistry common driver and databases (QCDB[274]) to be interoperable with the NWChem, PSI4, and CFOUR programs. QCDB is written in Python.

GAMESS has been interfaced with QCDB in such a way that one can generate input files for PSI4, NWChem, CFOUR, and GAMESS using a common input syntax. These input files are user-friendly and easy to use even for beginners. For example, an input file to calculate the MP2/cc-pVTZ energy of the water molecule is as simple as

```
h2o = qcdb.set_molecule("""
O
H 1 1.8
H 1 1.8 2 104.5
units au
""")
qcdb.set_options({'basis':'cc-pVTZ'})
qcdb.energy('gms-mp2')
```

In the above notation, 'gms' stands for GAMESS. One can do the above computation in NWChem, PSI4, and CFOUR by changing the 'gms-mp2' to 'nwc-mp2', 'p4-mp2', or 'c4-mp2', respectively. In addition, QCDB can parse the output of those programs to produce a common output. The flexibility of the input and output format reduces extra effort for users to execute programs and manage data seamlessly, regardless of the program.

The most beneficial part of the GAMESS-QCDB interface is that users are able to perform *ab initio* electronic structure calculations across multiple programs, taking advantage of the unique features of each program. For example, one can perform a very high-level benchmark computation on a molecular cluster using the CCSD(T)/cc-pV[Q5]Z level in Psi4 and then do a post-CCSD(T) correction using NWChem or corrections computed via the EFMO approach, for larger molecular clusters, in GAMESS. GAMESS–QCDB is beneficial to other programs as well. For example, the GAMESS interface provides the EFP capability through the GAMESS potential file generation (MAKEFP) and then running EFP calculations on molecular clusters for the other programs.

There are some methods, of course, such as HF, DFT, MP2, and coupled-cluster methods that are common to the GAMESS, NWChem, PSI4, and CFOUR programs. However, there are unique features in each program as well.

IV. MODERN PROGRAMMING PRACTICES

As high performance computing enters the exascale era, new paradigms must be adopted. This is especially true for widely used electronic structure packages such as GAMESS. In addition to the development of strategies for parallel computer coding, some of which have been discussed in previous sections, consideration must be given to the power consumption by massively parallel computers (i.e., Dennard's law[275]), which can be as costly on an annual basis as the initial cost of the hardware. This means that strategies are needed for minimizing the power consumption while at the same time optimizing the time to solution. An equally important consideration is how to optimize the development, testing, and distribution of codes that are increasingly complex. These issues are discussed in Secs. IV A and IV B.

A. Maximizing performance under power constraints

Energy consumption has become a major design constraint in modern computing systems for which a power envelope has been established between 20 MW and 40 MW. Hence, GAMESS scaling capabilities have to take into account the efficient usage of the available power allocation, in addition to the efficiency of calculations. A way to achieve efficient power usage has been implemented in GAMESS such that the operating core frequency and voltage are reduced to the minimum[276] for the cores hosting the data-servers because they do not participate in power demanding (computational) tasks. Recently, power allocation strategies among DRAM, GPU, and CPU have been proposed for the hybrid CPU–GPU LibCChem implementation (See Sec. V) that targets full GPU utilization, and thus, the GPU may require high priority in power utilization. Previous experiments, however, showed that the highest priority should be given to DRAM if (part of) a calculation is memory-intensive, such as storing/reading the integrals, to avoid a huge performance penalty.[277] Then, the GPU gets the second priority for allocating power to maximize the performance of GPU-intensive phase

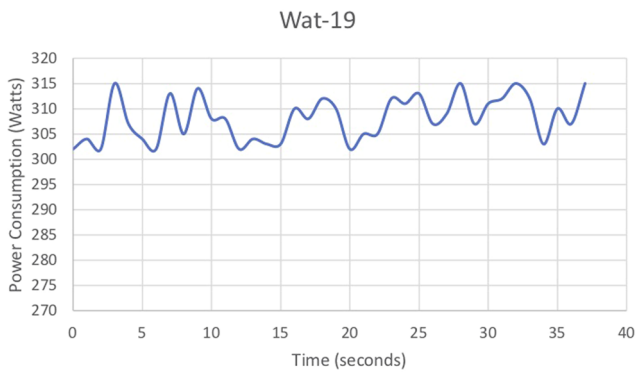

FIG. 2. The power consumption of the wat-19 calculation using the feedback strategy when the power allocation is set to 315 W. The power shown is the sum of the instantaneous GPU, CPU, and DRAM power, which are obtained from nvidia-smi for GPU and from the Intel Running Average Power Limit (RAPL) interface for CPU and DRAM.

of the application. Within the CPU power domains, the remaining power budget is allocated in accordance with GAMESS calculation performance at runtime.[278] To determine the amount of power to be allocated to the GPU, a feedback strategy is employed based on current utilization of GPU components gathered using the NVIDIA system management interface.

When power is allocated as per the above strategy, experiments on a 28-core Haswell-EP platform equipped with a Kepler K40m GPU with five different GAMESS/Libcchem calculations showed that the strategy provided maximum performance even with reduced power consumption. Specifically, an 11% reduction in power consumption did not reduce performance at all, while a 17% reduction in power resulted in only a 2% performance loss. Figure 2 shows the Libcchem power usage during an HF calculation of a 19-water molecule cluster (Wat-19), when the total power budget was set to 315 W (11% of the 351 W used with maximum power needed).

In summary, power re-allocation strategies have been successfully used in GAMESS Libcchem to improve its energy efficiency.

B. Software development process

1. Version control and source repository

The GAMESS development source code is hosted on the GitHub collaborative development platform[279] within a private repository and is managed using the git[280] distributed version control system. GAMESS employs the gitflow branching model[281] with separate dedicated branches for development and releases. Forking is disabled to ensure that all development undergoes continuous integration, a software-engineering best practice of building and testing every code change committed to a shared repository. With respect to the gitflow branching model, all branches in GAMESS with new commits undergo integration testing. Daily testing is performed for the development and releases branches to ensure that those branches are always in a stable state. More information regarding how to contribute to the development of GAMESS may be found at: https://www.msg.chem.iastate.edu/gamess/developers.html.

2. Continuous integration

GAMESS utilizes two continuous integration platforms for integration testing: Travis[282] and Jenkins.[283] Travis is a cloud-based continuous integration platform. For open source and academic research codes, Travis provides access to a single cloud-based compute instance for building and testing known as a worker. The GAMESS Travis worker is configured to perform 5 build-tests on a 64-bit instance running Ubuntu 14.04.5 LTS that varies the communication model (e.g., sockets and MPICH MPI[284]), math library (e.g., ATLAS[285] and netlib[286]), and build option (e.g., non-threaded and OpenMP-threaded GAMESS). For one of the build-tests, the FTNCHEK static analyzer is used to analyze the Fortran code for issues such as common block alignment, variable usage before initialization, and code formatting. A summary of the GAMESS Travis worker build-test configuration is provided in Table III.

Each build-test compiles GAMESS using the GNU compiler and performs validation testing using a small test set consisting of serial and parallel runs. Although free and in the cloud (off-premise), the Travis continuous integration platform has many limitations such as available compiler and hardware support (e.g., no access to GPGPUs) and testing restrictions (e.g., worker time-out if no output is received for any 10-min time period).

To address the limitations of the Travis continuous integration platform, an on-site installation of the Jenkins continuous integration platform was deployed and interfaced with local computing resources to facilitate additional build-test configurations. For pull-requests (code-integration requests) into the development branch, Jenkins performs 6 build-tests in parallel running 64-bit Centos 7 that varies the compiler (e.g., GNU,[287] Intel,[288] and PGI[289]), communication model (e.g., sockets, OpenMPI,[290] and Intel MPI[291]), and math library (e.g., OpenBLAS,[292] Intel MKL,[293] and PGI BLAS[289]).

TABLE III. GAMESS Travis worker build-test configuration.

			Tests		
Math	Comm.	Build option	Build	Validation	Static analysis
ATLAS	Sockets	Non-threaded	Yes	Yes	Yes
ATLAS	MPICH MPI	Non-threaded	Yes	Yes	No
Netlib	Sockets	Non-threaded	Yes	Yes	No
Netlib	Sockets	Non-threaded	Yes	Yes	No
ATLAS	MPICH MPI	Threaded	Yes	Yes	No

TABLE IV. GAMESS Jenkins build-test configuration for integration into the development branch.

Compiler	Math	Comm.	Tests	
			Build	Validation
GNU	OpenBLAS	Sockets	Yes	Yes
GNU	OpenBLAS	OpenMPI	Yes	Yes
Intel	Intel MKL	Sockets	Yes	Yes
Intel	Intel MKL	Intel MPI	Yes	Yes
PGI	Intel MKL	OpenMPI	Yes	Yes
PGI	PGI BLAS	OpenMPI	Yes	Yes

All six build-tests are performed on the same CPU architecture. Each build-test compiles the non-threaded version of GAMESS and performs validation testing using a large test set consisting of 665 serial and 529 parallel runs with a test coverage of over 60% measured using gcov.[287] A summary of the GAMESS Jenkins build-test configuration is provided in Table IV.

A smaller set of appropriate build-tests configurations are performed for LibCChem and OpenMP threaded development.

For pull-requests into the release branch (e.g., new scheduled public release), multiple computing architecture (e.g., Intel Sandybridge, Intel Haswell, Intel Skylake, AMD EPYC, and NVIDIA GPGPU) testing is performed using the GNU compiler. Each CPU architecture consists of four build-tests for the non-threaded build of GAMESS. The OpenMP threaded build of GAMESS and the LibCChem CPU-only build consist of two build-tests each. The GPU-accelerated LibCChem build of GAMESS currently consists of a single build-test using an NVIDIA GPGPU (e.g., K20, K40, K80, or V100). A summary of the GAMESS Jenkins build-test configurations for multiple computing architectures is provided in Table V.

The total wall-time to complete all Jenkins build-tests in Table III is ~72 h and is dependent on resource availability. Due to the heavy load placed on local computational resources, Jenkins can only be triggered by a successful Travis session.

3. Testing framework

The GAMESS testing framework consists of a set of Python scripts that provides the functionality of running GAMESS inputs and parsing and validating the generated output. The Python parse takes any GAMESS log file and extracts all predefined content and stores the content into a validation file in JSON object notation. During testing, a generated log file is parsed and a similar JSON object file is created containing the parsed values. The name of the log file and the number of validation entries must match in order for validation to proceed. For each validation entry, the values are compared and measured against the specified tolerance for each entry.

The testing framework is designed to work with unstructured output commonly encountered when using scientific software. The parsed content can be extended to accommodate new unstructured output by defining new parse groups.

4. Portability and source code

GAMESS prides itself in being a highly portable quantum chemistry code. End-users have the option of compiling from source, using pre-compiled binaries, or obtaining a Singularity container image.

GAMESS can be compiled with minimal third-party dependences on many variants of 32-bit and 64-bit Linux, Apple, and Microsoft Windows operating systems. At minimum, GAMESS requires the C-shell, a C and Fortran compiler, and the GNU make tool. The GAMESS build process is coordinated using several C-shell scripts. These C-shell scripts have recently been integrated with the GNU make tool to enable parallel compilation of the Fortran sources files. The build process involves invoking a C-shell script, config, which will prompt the end-user to provide the build target, build directory location, binary name, compiler choice, math library selection, communication mode (e.g., sockets or MPI), and additional build options (e.g., with the Michigan State Coupled-Cluster Theory package, with OpenMP threading, or with LibCChem). The latter process will generate a file, install.info, containing the build configuration.

Pre-compiled GAMESS binaries are made available for 64-bit Microsoft Windows users. These native binaries are prepared using the latest PGI Community Edition compilers and provide end-users

TABLE V. GAMESS Jenkins build-test configuration for integration into the release branch.

Architecture	Math	Comm.	Build option	Tests	
				Build	Validation
Intel Sandybridge	Netlib	Socket	Non-threaded	Yes	Yes
Intel Haswell	Netlib	OpenMPI	Non-threaded	Yes	Yes
Intel Skylake	OpenBLAS	Socket	Non-threaded	Yes	Yes
AMD EPYC	OpenBLAS	OpenMPI	Non-threaded	Yes	Yes
AMD EPYC	Intel MKL	OpenMPI	Threaded	Yes	Yes
Intel Skylake	Intel MKL	OpenMPI	Threaded	Yes	Yes
AMD EPYC	Intel MKL	OpenMPI	LIBCCHEM CPU-only	Yes	Yes
Intel Skylake	Intel MKL	OpenMPI	LIBCCHEM CPU-only	Yes	Yes
Intel Sandybridge + NVIDIA GPGPU	Intel MKL	OpenMPI	LIBCCHEM	Yes	Yes

with two options for statically linked math libraries (e.g., Intel MKL or PGI BLAS). The only supported distributed data interface (DDI) communication mode for Microsoft Windows is MPI; therefore, end-users are required to install the Microsoft MPI library (MS-MPI) provided with the pre-compiled GAMESS binary package. Several Windows batch scripts, shortcuts, and step-by-step visual instructions are provided with the pre-compiled GAMESS binary package to help lower the barrier to learning how to deploy GAMESS using the Windows Command Prompt.

5. Singularity container

Alternatively, through an agreement between GAMESS and NVIDIA, end-users have the option of deploying GAMESS using a GPU-enabled Singularity[294] container image.[295] The GAMESS container image provided by NVIDIA contains a pre-configured and pre-installed copy of GAMESS with the GPU-accelerated LibCChem package. The GPU-enabled container image of GAMESS was made available on the NVIDIA GPU Compute Cloud (NGCC) during its inaugural launch at the Supercomputing 2017 conference.[296] Application containers are emerging technology, which have the potential to help improve code portability and reproducibility in scientific computing. Scientific software developers can use application containers to package and share images of their pre-configured and pre-installed application along with all software dependences. For scientific software with complex software dependences, such as the GPU-accelerated LibCChem package (e.g., CUDA,[297] Global Arrays,[298] BLAS, LAPACK,[299] and HDF5[300]), application containers can significantly lower the barrier to usage. As application container technology becomes more available across operating system vendors, GAMESS may consider offering a single container image as a cross-platform solution.

V. LibCChem

The breakdown of Dennard[301] scaling marked the beginning of a new computational era in which the familiar latency-oriented processor architectures were (almost completely) replaced by throughput-oriented ones for performance purposes. Among the latter, during the last two decades, GPUs[302] have cemented their status as near-ideal "number-crunching" machines, delivering the lion's share of the FLOP performance achieved by the most powerful supercomputers in the world. For example, Summit, the fastest supercomputer in the world according to the Top500, acquires 95% of its peak FLOP performance from its NVIDIA V100 GPUs.[303]

As computational hardware morphs into these novel, intrinsically parallel architectures, quantum chemical methods and their underpinning implementations must evolve accordingly. GAMESS[1] has started on this evolution via its use of LibCChem,[14–16,304] a specialized C++ library designed for high-performance electronic structure theory computing on both CPUs and GPUs. Besides introducing object-oriented programming into GAMESS, LibCChem also enables GPU usage through its use of the CUDA programming model for execution on NVIDIA graphics cards. Currently, LibCChem can execute a number of different quantum chemistry calculations on NVIDIA GPUs. These include the evaluation of electron-repulsion integrals (ERIs) via the Rys quadrature algorithm,[14] the Fock build step of Restricted Hartree–Fock (RHF) energy calculations,[15] the evaluation of MP2 energies,[304] and the calculation of RI-MP2 energies and gradients.

Recently, also a new GPU port path has been enabled directly within GAMESS itself via the usage of OpenMP GPU offloading of the hybrid MPI/OpenMP RI-MP2 (Fortran) code.[102]

In Secs. V A–V E, we discuss the GPU implementations in LibCChem and GAMESS Fortran and their performance.

A. Integrals

The evaluation of integrals is arguably the most common bottleneck in quantum chemistry.[305,306] For this reason, efficient integral evaluation has been a historically prolific area of research[307–331] leading to algorithmic enhancements that have been instrumental in enabling quantum chemical calculations on increasingly large systems. In this section, we will focus on the GAMESS capability to evaluate ERIs on GPUs. Their sheer number—formally $O(N^4)$—makes their computation the most expensive step of an SCF procedure and an obvious candidate for accelerator offloading.

1. Rys quadrature

The first algorithm for the evaluation of ERIs on GPUs in LibCChem was implemented in 2010 by Asadchev *et al.*[14] The code, which is still operational, provides a high-performance Rys quadrature algorithm in which memory access patterns and data reuse were specifically optimized for GPUs. In the Rys quadrature scheme, ERIs are evaluated as combinations of 2D-integrals (Rys integrals), which are largely shared among different ERIs within a given integral class. The LibCChem implementation maps each ERI class to a different thread block. The Rys integrals are stored on a per-block basis in the shared memory of the GPU, enabling their efficient reuse (within a thread block) when constructing an ERI class. To further improve performance, the Rys quadrature implementation in LibCChem was designed to have two execution modes: a small and a large angular momentum path. Within the small angular momentum path, Rys integrals are evaluated using polynomial expressions obtained by fully expanding the recurrence relations. These formulas were then parsed through Sage,[332] a Python package that performs Common Subexpression Elimination (CSE). This way, the polynomial expressions were simplified and reordered to maximize register reuse. For high angular momentum classes, the 2D-integrals are instead evaluated, as traditionally, in terms of recursion, and transfer relations and then combined to form the ERIs. The Rys quadrature integral code was later coupled with a novel Hartree–Fock algorithm, which presented a maximum speed-up of 38.9× against the GAMESS/Rys implementation. The GPU version of the algorithm showed a single-core maximum speed-up of 17× against the CPU version (*vide infra*).

2. LibAccInt

Currently, LibCChem can also evaluate ERIs via the Head–Gordon–Pople (HGP) method due to the interface with the LibAccInt ERI library. LibAccInt (Library for Accelerated Integral evaluation) is a standalone integral library that is intended to interface with any quantum chemistry code. However, special consideration is given to the GAMESS FORTRAN code and the LibCChem C++ code. An interface to connect GAMESS and LibCChem with LibAccInt is in progress, in order to enable optimized integral routines

targeted for GPU execution, while also providing parallel CPU evaluation. The library will support several GPU programming models, in order to execute on GPU architectures. The initial implementation of the library is based on the CUDA execution model. The initial algorithm in planning for the library is the Head–Gordon–Pople[331] (HGP) algorithm, an optimized version of the Obara–Saika[307] algorithm and an excellent approach for mid-contraction degrees. Additionally, the Obara–Saika algorithm, which outperforms the HGP algorithms for certain contraction degrees and combinations of angular momenta, is planned for implementation. Finally, early contraction schemes for the fast evaluation of highly contracted low-angular-momentum ERIs, such as the Pople–Hehre[309] axis switch method and the CCTTT path in the PRISM algorithm,[310] will also be implemented. The integration of this GPU oriented library will enable GAMESS and GAMESS + LibCChem to possess extremely fast and efficient routines for integral calculations.

B. HF, GFB

The formation of the two-electron portion of the Fock matrix in the Fock build step is the most computationally expensive procedure in a Hartree–Fock (HF) implementation. The Fock build step is composed of two main algorithmic stages: (i) the evaluation of the ERIs and (ii) the contraction of the computed ERIs with the corresponding density matrix elements, which are then added into the Fock matrix. LibCChem offloads both stages to GPUs, effectively minimizing the Fock build runtime.[15]

The goal for the new HF implementation was twofold: (1) to create a high-performance HF code for medium-sized systems (i.e., on the order of ~2000–3000 basis functions) and (2) to create a HF code that minimizes, if not entirely eliminates, the need for required synchronization between threads.

The new HF code uses a number of algorithmic design features to achieve its two goals. The first design feature is its distribution of tasks to different GPUs via a *binned shell-pair container*. The binned shell pair container is a three-dimensional container that contains batches of shell pairs such that all shell pairs in a batch are from the same shell pair class. The shell pair container arranges these shell pair batches in two ways—by shell pair class cost and by shell pair value, where the shell pair value refers to the exponent of the largest integral that can be calculated by the shell pair batch. The use of a binned shell pair container achieves two goals. First, screening can automatically be performed through a smart combination of shell pair batches to create shell quartet batches. This is because the value of the largest integral created by a shell quartet batch can be uniquely mapped to the sum of the shell pair value of the two constituent shell pair batches. Thus, screening can occur at the task distribution stage rather than after task distribution, significantly improving load balance. Additionally, all shell quartets within a shell quartet batch will have exactly the same computational cost, leading to a perfectly load-balanced computation of the ERIs arising from a shell quartet batch.

The shell quartet batches are formed by combining a bra shell pair batch with one or multiple ket shell pair batches, and subsequently distributed to different MPI ranks (i.e., GPUs) via a master-slave model. The cost of each shell pair batch is calculated beforehand, and the most expensive shell quartet batches are formed first and distributed first, efficiently balancing the workload across the active processes. Within a shell quartet batch, the different shell quartets are distributed to different GPU threads so that the ERI computation is performed for each shell quartet by a single thread. As each shell quartet in a shell quartet batch will have the same code path, this enables taking advantage of the SIMT hardware architecture model that GPUs offer.

To minimize the required thread synchronization, a novel Fock contraction has been devised and implemented. First, multiple portions of the Fock matrix are stored and written to separately. The first such portion is the J (Coulomb) matrix, which contains the two Coulomb blocks used in the Fock contraction step. For a given shell quartet batch, these are written completely in parallel without any thread synchronization. The exchange (K) portions are written to via the use of a series of three-dimensional exchange (K) arrays. For a given shell quartet batch, there exists one K array per exchange block, leading to four K arrays being written to overall. Each K array is a three-dimensional object, where the first two dimensions represent a location on the Fock matrix and the third dimension represents a position within a buffer to be flushed to a given Fock matrix element determined by the first two dimensions. During the computation of the exchange elements in the Fock contraction step, each exchange element is written to a unique location in one of the four K arrays, eliminating the need for any thread synchronization. After the K arrays are written to, they are flushed in parallel into the Fock matrix. In this way, the new HF code achieves a minimal amount of required thread synchronization, as both the Coulomb and exchange elements can be added to the Fock matrix with no thread synchronization whatsoever.

The benefits of such an approach can be seen in the speed-up of the new LibCChem HF code compared to the default GAMESS MPI-parallel HF code on Summit. Both codes were run on a 150-water cluster using the PC0 segmented basis set. The LibCChem code was run using a single V100 GPU. The CPU code was run in parallel using 3, 6, 9, 12, 15, 18, and 21 threads on a single Summit Power9 CPU. This was done to compare the performance of the new LibCChem code against a parallel run of the GAMESS HF code and also to compare the performance of the new LibCChem HF code against the predicted serial timing of the GAMESS HF code, determined via extrapolation. Compared to the fully parallel GAMESS HF code using 21 threads, that is, using the 21 cores of the P9 processor at Simultaneous Multi-Threading (SMT) level 1, the new LibCChem code achieved a speed-up of ~39×. Against the predicted serial timing of the GAMESS HF code, the new LibCChem HF algorithm achieved a speed-up of ~755×. These significant speed-ups display both the effectiveness of the current LibCChem HF algorithm and the effectiveness of GPUs as a "number-crunching" machine.

C. MP2

Along with ERI computations and RHF Fock build calculations, LibCChem can also perform MP2 energy calculations on GPUs. The initial implementation of LibCChem MP2 was written by Tomlinson *et al.*,[304] initially devised for CPUs, and later ported to GPUs. This implementation brought a number of memory footprint and performance improvements, when compared to the original GAMESS implementation. First, chained matrix operations were reordered to minimize the FLOP count, and highly

tuned math libraries were adopted for all of the integral transformations. Second, the per-core memory footprint was significantly reduced, thereby enabling computations using several thousands of basis functions. Third, the I/O bottleneck was alleviated by implementing a strategy that uses OpenMP to assign threads to some I/O duties, while other threads are performing compute operations. The use of the HFD5 parallel file storage library was used to further enhance I/O.

D. RI-MP2

The resolution of the identity (RI) approximation is a way to ameliorate the cost of the evaluation of ERIs, as discussed in Sec. III A. The RI approximation factorizes the four center two electron integrals into the product of two and three center integrals, which significantly reduces the cost. It can be applied to any method that evaluates ERIs. The first implementation of the RI methodologies in LibCChem was for the MP2 method. Two sets of RI-MP2 methods are found within GAMESS + LibCChem: a hybrid MPI/OpenMP CPU version (Fortran), which now supports OpenMP GPU offloading,[102] and a CUDA version in LibCChem that supports NVIDIA GPUs.

The RI-MP2 implementations in GAMESS and LibCChem focus the GPU intensive tasks in the dense matrix multiply operations needed for the MO and the $V^{-1/2}$ transformation and to create the four-center two-electron integrals from the three-center ones. This is done via cuBLAS enabled matrix operations on GPUs. In GAMESS, the final RI three center integrals can be stored in three levels using (i) distributed memory, which can be expanded to a desired size by adding more compute nodes; (ii) CPU compute node memory, which is a fixed number and usually varies in the range of ~250 GB to 512 GB for modern multicore CPUs; and (iii) GPU high bandwidth global memory, which is as small as ~16 GB. In the fragmentation context, the whole matrix can usually fit on CPU node memory and/or the GPU global memory. For large calculations, the matrix storage can be spilled to the distributed memory. Benchmark calculations for water clusters and fullerene showed that the speed-up of the GPU RIMP2 energy kernel using a single V100 GPU relative to the MPI/OpenMP RI-MP2 energy code using a P9 socket (21 physical cores, 4 hardware threads) is 14× .[102] This has demonstrated that directive-based offloading implementations can perform near the GPU/CPU theoretical speed-up based on the machine peak ratios.

In LibCChem, the storage of the necessary quantities, such as the integrals, is done in the AO basis. If the integrals to store are small, the RI-MP2 algorithm will use the Global Arrays toolkit to store them. However, if they are large the HDF5 parallel I/O library is used. In benchmark calculations conducted on a 150-H_2O cluster using the aug-cc-pVDZ basis set, the LibCChem RI-MP2 algorithm yielded a speed-up of ~14× using a single V100 NVIDIA GPU compared to the (LibCChem RI-MP2) CPU code run on a P9 socket using 21 cores at SMT level 2 (increasing the SMT from 2 to 4, did not yield any CPU performance improvement). The speed-up of a calculation on the same molecular system using 66 V100 GPUs was 798× with respect to the fully parallel execution on the P9 CPU. This showed that the RI-MP2 LibCChem implementation can reach near the V100 peak throughput on a single GPU, maintaining also an extremely high performance when operating on a large number of GPUs.

E. Usability

Throughout its lifespan, LibCChem has seen significant improvements to its usability as a library. The goal of such enhancements was to meet the following design features: (i) the code must be maintainable and readable, allowing for further development; (ii) it should be modular, enabling users to build only selected functionalities; (iii) it should always be compatible with modern versions of its critical dependences; and (iv) it must be designed to be highly portable.

General improvements in the usability of LibCChem came in different forms. At its inception, LibCChem was built using a patched version of GNU Autotools, the Boost library, and the Global Arrays library, making it extremely complicated to build in any other system than the one it was developed on. For this reason, the build system was replaced with CMake, a modern, cross-platform, open source tool for managing the building of packages. Transitioning to a modern build system and not relying on a patched version of Autotools enabled LibCChem to update to the most modern versions of the Boost and Global Arrays libraries.

Another major improvement to LibCChem usability was achieved by upgrading from the C++98 standard to C++11. Such a modernization in the standard adopted by the library resulted in fewer external dependences and in the capability to access modern C++ constructs. This also lowered the reliance of LibCChem on the Boost library, as many of its functionalities are directly provided by the C++11 standard. LibCChem was also made more modular. The library is capable of performing different types of calculations. However, originally these different functionalities were not modularized. All of the available methods were by default compiled and linked to GAMESS. To streamline both the build and development processes, the code has been split into four "modules"—the Hartree–Fock module, the RI module (containing both RI-MP2 and density-fitted Hartree–Fock), the CC module, and the MP2 module. These modules can now be selected for compilation during the configuration of GAMESS. During the LibCChem build process, the build system recognizes, which modules the user would like to build and builds only the files associated with those modules.

Along with build system improvements, LibCChem compiler support was augmented. LibCChem now supports a wide variety of compilers for compilation usage. Specifically, the library has been tested and proven to work on the GCC (up to 9.1), Intel, PGI, XL, and Clang compiler tool chains. Additionally, LibCChem has been tested and shown to compile using GCC on ARM and IBM architectures. This increase in compiler support was facilitated primarily via the removal of non-portable code from LibCChem such as the "pedantic" warnings issued by the GCC compiler with the -pedantic flag. This compiler flag issues warnings for any code that does not strictly conform to the ISO C standard. Generally, this consists of code that uses GCC-specific extensions and would thus not compile on other compilers.

Started in the early 2010s, the LibCChem library has served as the GPU arm of GAMESS. With GPUs becoming more and more important in the world of high-performance computing, LibCChem has become a more significant part of the GAMESS software package. For this reason, LibCChem has seen many improvements since its inception, ranging from algorithmic changes to upgrades in usability by users of the library.

Along with LibCChem, however, interest has also risen in directly offloading GAMESS calculations to GPUs via the use of OpenMP. GPU offloading via a pragma-based approach allows for portability, as the standard could support multiple vendors within it. At this point, the GPU offloading of GAMESS Fortran is limited by compiler support. Future development of GAMESS in this area is to come.

VI. SUMMARY AND FUTURE

GAMESS is a broad-based multi-function living electronic structure code. Many of the future developments of GAMESS and Libcchem have already been mentioned. These include the development of fully analytic gradients for the QM–EFP2 method and for the CCSD(T) and CR-CC(2,3) coupled cluster methods, and the development of RI-CC methods and their integration with the FMO and EFMO fragmentation methods. Fully analytic EFMO gradients are almost, but not quite completed, and the derivation and coding of fully analytic gradients for the AFO version of FMO are in progress. The development of highly parallel codes is planned for all of the coupled cluster methods that have been implemented in GAMESS and that will be implemented in LibCChem. The various components in Libcchem, such as LibAccInt, the generalized Fock build, and RI-MP2, will be more seamlessly integrated. In all of these endeavors, improving the parallelism and overall computational efficiency will be a central focus.

Gagliardi, Truhlar, and co-workers have developed the multi-configurational pair density functional theory (MCP-DFT) that introduces multi-configurational character into DFT. Their implementation will soon be released in GAMESS. Shortly thereafter, the MCP-DFT analytic gradients will be added.

Work is underway on enriching the existing CC routines with the double electron-attachment and double ionization potential EOMCC options, which are particularly useful in determining the electronic spectra of biradicals,[154] and approximate coupled-pair approaches, which extend traditional CC truncations to a strongly correlated regime.

ACKNOWLEDGMENTS

The development of GAMESS was supported by several agencies, including the Air Force Office of Scientific Research, the Department of Defense, the Department of Energy, the National Science Foundation, and the National Institutes of Health and was also supported by Intel, Microsoft, and NVIDIA. The members of the GAMESS development group are very grateful for this support. The development of GAMESS has benefited from the support of many supercomputer centers, including those sponsored by the National Science Foundation (e.g., NCSA), the Department of Defense via the High Performance Computing Modernization Program, and the Department of Energy (e.g., the Argonne Leadership Computing Facility, the Oak Ridge Leadership Computing Facility, and the National Energy Research Scientific Computing Center). The authors would like to thank Dr. Takeshi Nagata whose contributions to GAMESS are numerous and who passed away at too young an age.

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

REFERENCES

[1] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, and J. A. Montgomery, J. Comput. Chem. **14**, 1347 (1993).
[2] M. S. Gordon and M. W. Schmidt, in *Theory and Applications of Computational Chemistry*, edited by C. E. Dykstra, G. Frenking, K. S. Kim, and G. E. Scuseria (Elsevier, Amsterdam, 2005), pp. 1167–1189.
[3] B. Njegic and M. S. Gordon, J. Chem. Phys. **125**, 224102 (2006).
[4] H. Li and M. S. Gordon, J. Chem. Phys. **126**, 124112 (2007).
[5] K. K. Baldridge and V. Jonas, J. Chem. Phys. **113**, 7511 (2000).
[6] D. M. Chipman, J. Chem. Phys. **124**, 224111 (2006).
[7] J. R. Shoemaker and M. S. Gordon, J. Phys. Chem. A **103**, 3245 (1999).
[8] L. Bytautas and K. Ruedenberg, J. Chem. Phys. **121**, 10852 (2004).
[9] N. Minezawa and M. S. Gordon, J. Phys. Chem. A **113**, 12749 (2009).
[10] N. Minezawa, N. De Silva, F. Zahariev, and M. S. Gordon, J. Chem. Phys. **134**, 054111 (2011).
[11] J. Mato and M. S. Gordon, Phys. Chem. Chem. Phys. **20**, 2615 (2018).
[12] A. C. West, M. W. Schmidt, M. S. Gordon, and K. Ruedenberg, J. Chem. Phys. **139**, 234107 (2013).
[13] R. M. Olson, J. L. Bentz, R. A. Kendall, M. W. Schmidt, and M. S. Gordon, J. Chem. Theory Comput. **3**, 1312 (2007).
[14] A. Asadchev, V. Allada, J. Felder, B. M. Bode, M. S. Gordon, and T. L. Windus, J. Chem. Theory Comput. **6**, 696 (2010).
[15] A. Asadchev and M. S. Gordon, J. Chem. Theory Comput. **8**, 4166 (2012).
[16] A. Asadchev and M. S. Gordon, J. Chem. Theory Comput. **9**, 3385 (2013).
[17] B. Q. Pham and M. S. Gordon, J. Chem. Theory Comput. **15**, 5252 (2019).
[18] B. Q. Pham and M. S. Gordon, J. Chem. Theory Comp. **16**, 1039 (2020).
[19] D. G. Fedorov, R. M. Olson, K. Kitaura, M. S. Gordon, and S. Koseki, J. Comput. Chem. **25**, 872 (2004).
[20] M. S. Gordon, Q. A. Smith, P. Xu, and L. V. Slipchenko, Annu. Rev. Phys. Chem. **64**, 553 (2013).
[21] S. R. Pruitt, C. Steinmann, J. H. Jensen, and M. S. Gordon, J. Chem. Theory Comput. **9**, 2235 (2013).
[22] J. Ivanic, J. Chem. Phys. **119**, 9364 (2003).
[23] W. Li, P. Piecuch, J. R. Gour, and S. Li, J. Chem. Phys. **131**, 114109 (2009).
[24] L. Roskop and M. S. Gordon, J. Chem. Phys. **135**, 044101 (2011).
[25] A. D. Findlater, F. Zahariev, and M. S. Gordon, J. Phys. Chem. A **119**, 3587 (2015).
[26] K. Kitaura, E. Ikeo, T. Asada, T. Nakano, and M. Uebayasi, Chem. Phys. Lett. **313**, 701 (1999).
[27] D. G. Fedorov and K. Kitaura, J. Chem. Phys. **120**, 6832 (2004).
[28] D. G. Fedorov and K. Kitaura, J. Phys. Chem. A **111**, 6904 (2007).
[29] D. G. Fedorov, T. Nagata, and K. Kitaura, Phys. Chem. Chem. Phys. **14**, 7562 (2012).
[30] D. G. Fedorov, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **7**, e1322 (2017).
[31] Y. Nishimoto and D. G. Fedorov, J. Chem. Phys. **148**, 064115 (2018).
[32] V. Mironov, A. Moskovsky, M. D'Mello, and Y. Alexeev, Int. J. High Perform. Comput. Appl. **33**, 212 (2019).
[33] S. R. Pruitt, H. Nakata, T. Nagata, M. Mayes, Y. Alexeev, G. Fletcher, D. G. Fedorov, K. Kitaura, and M. S. Gordon, J. Chem. Theory Comput. **12**, 1423 (2016).
[34] D. G. Fedorov, Y. Sugita, and C. H. Choi, J. Phys. Chem. B **117**, 7996 (2013).
[35] S. Ito, D. G. Fedorov, Y. Okamoto, and S. Irle, Comput. Phys. Commun. **228**, 152 (2018).
[36] T. Nagata, K. Brorsen, D. G. Fedorov, K. Kitaura, and M. S. Gordon, J. Chem. Phys. **134**, 124115 (2011).
[37] H. Nakata and D. G. Fedorov, Phys. Chem. Chem. Phys. **21**, 13641 (2019).
[38] H. Nakata, D. G. Fedorov, S. Yokojima, K. Kitaura, M. Sakurai, and S. Nakamura, J. Chem. Phys. **140**, 144101 (2014).
[39] D. G. Fedorov, A. Brekhov, V. Mironov, and Y. Alexeev, J. Phys. Chem. A **123**, 6281 (2019).
[40] D. G. Fedorov and K. Kitaura, J. Chem. Phys. **147**, 104106 (2017).

[41] D. S. Kaliakin, D. G. Fedorov, Y. Alexeev, and S. A. Varganov, J. Chem. Theory Comput. **15**, 6074 (2019).
[42] T. Nagata, D. G. Fedorov, T. Sawada, and K. Kitaura, J. Phys. Chem. A **116**, 9088 (2012).
[43] D. G. Fedorov, J. Chem. Theory Comput. **15**, 5404 (2019).
[44] D. G. Fedorov and K. Kitaura, J. Phys. Chem. A **122**, 1781 (2018).
[45] H. Nakata, Y. Nishimoto, and D. G. Fedorov, J. Chem. Phys. **145**, 044113 (2016).
[46] H. Nakata, T. Nagata, D. G. Fedorov, S. Yokojima, K. Kitaura, and S. Nakamura, J. Chem. Phys. **138**, 164103 (2013).
[47] M. C. Green, H. Nakata, D. G. Fedorov, and L. V. Slipchenko, Chem. Phys. Lett. **651**, 56 (2016).
[48] H. Nakata, D. G. Fedorov, S. Yokojima, K. Kitaura, and S. Nakamura, Chem. Phys. Lett. **603**, 67 (2014).
[49] D. G. Fedorov and K. Kitaura, J. Chem. Phys. **122**, 054108 (2005).
[50] T. Ikegami, T. Ishida, D. G. Fedorov, K. Kitaura, Y. Inadomi, H. Umeda, M. Yokokawa, and S. Sekiguchi, J. Comput. Chem. **31**, 447 (2010).
[51] T. Nagata, D. G. Fedorov, K. Ishimura, and K. Kitaura, J. Chem. Phys. **135**, 044110 (2011).
[52] H. Nakata, D. G. Fedorov, K. Kitaura, and S. Nakamura, Chem. Phys. Lett. **635**, 86 (2015).
[53] D. G. Fedorov and K. Kitaura, J. Chem. Phys. **123**, 134103 (2005).
[54] S. R. Pruitt, D. G. Fedorov, and M. S. Gordon, J. Phys. Chem. A **116**, 4965 (2012).
[55] K. R. Brorsen, F. Zahariev, H. Nakata, D. G. Fedorov, and M. S. Gordon, J. Chem. Theory Comput. **10**, 5297 (2014).
[56] M. Chiba, D. G. Fedorov, T. Nagata, and K. Kitaura, Chem. Phys. Lett. **474**, 227 (2009).
[57] D. G. Fedorov, T. Ishida, and K. Kitaura, J. Phys. Chem. A **109**, 2638 (2005).
[58] A. Stone, *The Theory of Intermolecular Forces* (Oxford University Press, 2013).
[59] P. N. Day, J. H. Jensen, M. S. Gordon, S. P. Webb, W. J. Stevens, M. Krauss, D. Garmer, H. Basch, and D. Cohen, J. Chem. Phys. **105**, 1968 (1996).
[60] I. Adamovic, M. A. Freitag, and M. S. Gordon, J. Chem. Phys. **118**, 6725 (2003).
[61] S. Yoo, F. Zahariev, S. Sok, and M. S. Gordon, J. Chem. Phys. **129**, 144112 (2008).
[62] P. Arora, L. V. Slipchenko, S. P. Webb, A. DeFusco, and M. S. Gordon, J. Phys. Chem. A **114**, 6742 (2010).
[63] M. Krauss and S. P. Webb, J. Chem. Phys. **107**, 5771 (1997).
[64] A. DeFusco, J. Ivanic, M. W. Schmidt, and M. S. Gordon, J. Phys. Chem. A **115**, 4574 (2011).
[65] A. DeFusco, N. Minezawa, L. V. Slipchenko, F. Zahariev, and M. S. Gordon, J. Phys. Chem. Lett. **2**, 2184 (2011).
[66] L. V. Slipchenko, J. Phys. Chem. A **114**, 8824 (2010).
[67] I. Adamovic and M. S. Gordon, J. Phys. Chem. A **109**, 1629 (2005).
[68] S. Sok, S. Y. Willow, F. Zahariev, and M. S. Gordon, J. Phys. Chem. A **115**, 9801 (2011).
[69] N. De Silva, M. A. Adreance, and M. S. Gordon, J. Comput. Chem. **40**, 310 (2019).
[70] E. B. Guidez and M. S. Gordon, J. Phys. Chem. A **121**, 3736 (2017).
[71] V. S. Bryantsev, M. S. Diallo, A. C. T. van Duin, and W. A. Goddard, J. Chem. Theory Comput. **5**, 1016 (2009).
[72] J. Řezáč, K. E. Riley, and P. Hobza, J. Chem. Theory Comput. **7**, 2427 (2011).
[73] J. H. Jensen and M. S. Gordon, Mol. Phys. **89**, 1313 (1996).
[74] H. Li, M. S. Gordon, and J. H. Jensen, J. Chem. Phys. **124**, 214108 (2006).
[75] D. D. Kemp, J. M. Rintelman, M. S. Gordon, and J. H. Jensen, Theor. Chem. Acc. **125**, 481 (2010).
[76] Q. A. Smith, K. Ruedenberg, M. S. Gordon, and L. V. Slipchenko, J. Chem. Phys. **136**, 244107 (2012).
[77] L. V. Slipchenko† and M. S. Gordon, Mol. Phys. **107**, 999 (2009).
[78] E. B. Guidez, P. Xu, and M. S. Gordon, J. Phys. Chem. A **120**, 639 (2016).
[79] T. Sattasathuchana, P. Xu, and M. S. Gordon, J. Phys. Chem. A **123**, 8460 (2019).
[80] M. W. Schmidt, E. A. Hull, and T. L. Windus, J. Phys. Chem. A **119**, 10408 (2015).
[81] M. A. Freitag, M. S. Gordon, J. H. Jensen, and W. J. Stevens, J. Chem. Phys. **112**, 7300 (2000).
[82] P. Xu, F. Zahariev, and M. S. Gordon, J. Chem. Theory Comput. **10**, 1576 (2014).
[83] L. Schoeberle, E. B. Guidez, and M. S. Gordon, J. Phys. Chem. A **122**, 6100 (2018).
[84] SAPT, 2020.
[85] B. Jeziorski, R. Moszynski, A. Ratkiewicz, S. Rybak, K. Szalewicz, and H. L Williams, "SAPT A Program for Many-Body Symmetry-Adapted Perturbation Theory Calculations of Intermolecular Interaction," 58 pp., http://www.physics.udel.edu/~szalewic/SAPT/SAPTtheo.pdf.
[86] C. Steinmann, D. G. Fedorov, and J. H. Jensen, J. Phys. Chem. A **114**, 8705 (2010).
[87] C. Bertoni and M. S. Gordon, J. Chem. Theory Comput. **12**, 4743 (2016).
[88] P. Pulay, Mol. Phys. **17**, 197 (1969).
[89] G. D. Fletcher, M. W. Schmidt, B. M. Bode, and M. S. Gordon, Comput. Phys. Commun. **128**, 190 (2000).
[90] J. L. Whitten, J. Chem. Phys. **58**, 4496 (1973).
[91] O. Vahtras, J. Almlöf, and M. W. Feyereisen, Chem. Phys. Lett. **213**, 514 (1993).
[92] F. Weigend, M. Häser, H. Patzelt, and R. Ahlrichs, Chem. Phys. Lett. **294**, 143 (1998).
[93] M. Häser and R. Ahlrichs, J. Comput. Chem. **10**, 104 (1989).
[94] Y. Alexeev, R. A. Kendall, and M. S. Gordon, Comput. Phys. Commun. **143**, 69 (2002).
[95] D. Fedorov and K. Kitaura, *The Fragment Molecular Orbital Method: Practical Applications to Large Molecular Systems* (CRC Press, 2009).
[96] M. S. Gordon, D. G. Fedorov, S. R. Pruitt, and L. V. Slipchenko, Chem. Rev. **112**, 632 (2012).
[97] K. Kitaura, T. Sawai, T. Asada, T. Nakano, and M. Uebayasi, Chem. Phys. Lett. **312**, 319 (1999).
[98] T. Nakano, T. Kaminuma, T. Sato, Y. Akiyama, M. Uebayasi, and K. Kitaura, Chem. Phys. Lett. **318**, 614 (2000).
[99] K. Ishimura, P. Pulay, and S. Nagase, J. Comput. Chem. **28**, 2034 (2007).
[100] C. M. Aikens, S. P. Webb, R. L. Bell, G. D. Fletcher, M. W. Schmidt, and M. S. Gordon, Theor. Chem. Acc. **110**, 233 (2003).
[101] M. Katouda and S. Nagase, Int. J. Quantum Chem. **109**, 2121 (2009).
[102] J. Kwack, C. Bertoni, B. Pham, and J. Larkin, WACCPD, 2019.
[103] P. Piecuch, S. A. Kucharski, K. Kowalski, and M. Musiał, Comput. Phys. Commun. **149**, 71 (2002).
[104] K. Kowalski and P. Piecuch, J. Chem. Phys. **113**, 18 (2000).
[105] K. Kowalski and P. Piecuch, J. Chem. Phys. **113**, 5644 (2000).
[106] P. Piecuch, S. A. Kucharski, and K. Kowalski, Chem. Phys. Lett. **344**, 176 (2001).
[107] P. Piecuch and M. Włoch, J. Chem. Phys. **123**, 224105 (2005).
[108] P. Piecuch, M. Włoch, J. R. Gour, and A. Kinal, Chem. Phys. Lett. **418**, 467 (2006).
[109] M. Włoch, J. R. Gour, and P. Piecuch, J. Phys. Chem. A **111**, 11359 (2007).
[110] J. Shen and P. Piecuch, J. Chem. Phys. **136**, 144104 (2012).
[111] J. Shen and P. Piecuch, Chem. Phys. **401**, 180 (2012).
[112] J. Hubbard, Proc. R. Soc. London, Ser. A **240**, 539 (1957).
[113] N. M. Hugenholtz, Physica **23**, 481 (1957).
[114] F. Coester, Nucl. Phys. **7**, 421 (1958).
[115] F. Coester and H. Kümmel, Nucl. Phys. **17**, 477 (1960).
[116] J. Čížek, J. Chem. Phys. **45**, 4256 (1966).
[117] J. Čížek, Adv. Chem. Phys. **14**, 35 (1969).
[118] J. Čižek and J. Paldus, Int. J. Quantum Chem. **5**, 359 (1971).
[119] J. Paldus, J. Čížek, and I. Shavitt, Phys. Rev. A **5**, 50 (1972).
[120] J. Paldus and X. Li, Adv. Chem. Phys. **110**, 1 (1999).
[121] R. J. Bartlett and M. Musiał, Rev. Mod. Phys. **79**, 291–352 (2007).
[122] J. A. Pople, R. Krishnan, H. B. Schlegel, and J. S. Binkley, Int. J. Quantum Chem. **14**, 545 (1978).
[123] R. J. Bartlett and G. D. Purvis, Int. J. Quantum Chem. **14**, 561 (1978).
[124] G. D. Purvis and R. J. Bartlett, J. Chem. Phys. **76**, 1910 (1982).

[125]K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, Chem. Phys. Lett. **157**, 479 (1989).
[126]P. Piecuch and K. Kowalski, *Computational Chemistry: Reviews of Current Trends* (World Scientific, 2000), pp. 1–104.
[127]M. Włoch, M. D. Lodriguito, P. Piecuch, and J. R. Gour, Mol. Phys. **104**, 2149 (2006); Erratum, **104**, 2991 (2006).
[128]P. Piecuch, K. Kowalski, I. S. O. Pimienta, and S. A. Kucharski, in *Low-Lying Potential Energy Surfaces; ACS Symposium Series*, edited by M. R. Hoffmann and K. G. Dyall (American Chemical Society, Washington, DC, 2002), Vol. 828, pp. 31–64.
[129]P. Piecuch, K. Kowalski, I. S. O. Pimienta, and M. J. McGuire, Int. Rev. Phys. Chem. **21**, 527 (2002).
[130]P. Piecuch, K. Kowalski, P.-D. Fan, and I. S. O. Pimienta, in *Advanced Topics in Theoretical Chemical Physics*, edited by J. Maruani, R. Lefebvre, and E. J. Brändas (Springer Netherlands, Dordrecht, 2003), pp. 119–206.
[131]P. Piecuch, K. Kowalski, I. S. O. Pimienta, P.-D. Fan, M. Lodriguito, M. J. McGuire, S. A. Kucharski, T. Kuś, and M. Musiał, Theor. Chem. Acc. **112**, 349 (2004).
[132]P. Piecuch, M. Włoch, and A. J. C. Verandas, in *Topics in the Theory of Chemical and Physical Systems*, Progress in Theoretical Chemistry and Physics Vol. 16, edited by S. Lahmar, J. Maruani, S. Wilson, and G. Delgado-Barrio (Springer, Dordrecht, 2007), pp. 63–121.
[133]S. A. Kucharski and R. J. Bartlett, J. Chem. Phys. **108**, 9221 (1998).
[134]P. Piecuch, J. R. Gour, and M. Włoch, Int. J. Quantum Chem. **108**, 2128 (2008).
[135]Y. Ge, M. S. Gordon, and P. Piecuch, J. Chem. Phys. **127**, 174106 (2007).
[136]Y. Ge, M. S. Gordon, P. Piecuch, M. Włoch, and J. R. Gour, J. Phys. Chem. A **112**, 11873 (2008).
[137]P. Piecuch, M. Włoch, and A. J. C. Varandas, Theor. Chem. Acc. **120**, 59 (2008).
[138]J. Shen and P. Piecuch, J. Chem. Theory Comput. **8**, 4968 (2012).
[139]P. Piecuch, S. A. Kucharski, and R. J. Bartlett, J. Chem. Phys. **110**, 6103 (1999).
[140]P. Piecuch, Mol. Phys. **108**, 2987 (2010).
[141]J. Noga and R. J. Bartlett, J. Chem. Phys. **86**, 7041 (1987); Erratum, **89**, 3401 (1988).
[142]J. F. Stanton and R. J. Bartlett, J. Chem. Phys. **98**, 7029 (1993).
[143]K. Kowalski and P. Piecuch, J. Chem. Phys. **120**, 1715 (2004).
[144]M. Włoch, J. R. Gour, K. Kowalski, and P. Piecuch, J. Chem. Phys. **122**, 214107 (2005).
[145]P. Piecuch, J. R. Gour, and M. Włoch, Int. J. Quantum Chem. **109**, 3268 (2009).
[146]G. Fradelos, J. J. Lutz, T. A. Wesołowski, P. Piecuch, and M. Włoch, J. Chem. Theory Comput. **7**, 1647 (2011).
[147]J. J. Lutz and P. Piecuch, Comput. Theor. Chem. **1040-1041**, 20 (2014).
[148]P. Piecuch, J. A. Hansen, and A. O. Ajala, Mol. Phys. **113**, 3085 (2015).
[149]J. D. Watts and R. J. Bartlett, Chem. Phys. Lett. **233**, 81 (1995).
[150]J. D. Watts and R. J. Bartlett, Chem. Phys. Lett. **258**, 581 (1996).
[151]O. Christiansen, H. Koch, and P. Jørgensen, J. Chem. Phys. **103**, 7429 (1995).
[152]J. R. Gour, P. Piecuch, and M. Włoch, J. Chem. Phys. **123**, 134113 (2005).
[153]J. R. Gour and P. Piecuch, J. Chem. Phys. **125**, 234107 (2006).
[154]J. R. Gour, P. Piecuch, and M. Włoch, Int. J. Quantum Chem. **106**, 2854 (2006).
[155]N. P. Bauman, J. A. Hansen, and P. Piecuch, J. Chem. Phys. **145**, 084306 (2016).
[156]N. P. Bauman, J. A. Hansen, M. Ehara, and P. Piecuch, J. Chem. Phys. **141**, 101102 (2014).
[157]J. Shen and P. Piecuch, J. Chem. Phys. **138**, 194102 (2013).
[158]K. Kowalski and P. Piecuch, J. Chem. Phys. **113**, 8490 (2000).
[159]K. Kowalski and P. Piecuch, J. Chem. Phys. **115**, 643 (2001).
[160]K. Kowalski and P. Piecuch, Chem. Phys. Lett. **347**, 237 (2001).
[161]K. Kowalski, S. Hirata, M. Włoch, P. Piecuch, and T. L. Windus, J. Chem. Phys. **123**, 074319 (2005).
[162]S. Li, J. Ma, and Y. Jiang, J. Comput. Chem. **23**, 237 (2002).
[163]S. Li, J. Shen, W. Li, and Y. Jiang, J. Chem. Phys. **125**, 074109 (2006).
[164]W. Li, P. Piecuch, and J. R. Gour, in *Advances in the Theory of Atomic and Molecular Systems: Conceptual and Computational Advances in Quantum Chemistry*, edited by P. Piecuch, J. Maruani, G. Delgado-Barrio, and S. Wilson (Springer Netherlands, Dordrecht, 2009), pp. 131–195.
[165]W. Li and P. Piecuch, J. Phys. Chem. A **114**, 6721 (2010).
[166]W. Li and P. Piecuch, J. Phys. Chem. A **114**, 8644 (2010).
[167]P. M. Kozlowski, M. Kumar, P. Piecuch, W. Li, N. P. Bauman, J. A. Hansen, P. Lodowski, and M. Jaworska, J. Chem. Theory Comput. **8**, 1870 (2012).
[168]W. Li, Z. Ni, and S. Li, Mol. Phys. **114**, 1447 (2016).
[169]J. Ivanic, J. Chem. Phys. **119**, 9377 (2003).
[170]J. Ivanic and M. W. Schmidt, J. Phys. Chem. A **122**, 5223 (2018).
[171]R. J. Gdanitz and R. Ahlrichs, Chem. Phys. Lett. **143**, 413 (1988).
[172]P. G. Szalay and R. J. Bartlett, Chem. Phys. Lett. **214**, 481 (1993).
[173]K. Hirao, Chem. Phys. Lett. **190**, 374 (1992).
[174]H. Li and J. H. Jensen, J. Comput. Chem. **25**, 1449 (2004).
[175]J. M. Rintelman, M. S. Gordon, G. D. Fletcher, and J. Ivanic, J. Chem. Phys. **124**, 034303 (2006).
[176]L. Bytautas and K. Ruedenberg, J. Chem. Phys. **121**, 10905 (2004).
[177]L. Bytautas and K. Ruedenberg, J. Chem. Phys. **121**, 10919 (2004).
[178]L. Bytautas and K. Ruedenberg, J. Chem. Phys. **122**, 154110 (2005).
[179]L. Bytautas and K. Ruedenberg, J. Chem. Phys. **124**, 174304 (2006).
[180]L. Bytautas, T. Nagata, M. S. Gordon, and K. Ruedenberg, J. Chem. Phys. **127**, 164317 (2007).
[181]L. Bytautas, N. Matsunaga, G. E. Scuseria, and K. Ruedenberg, J. Phys. Chem. A **116**, 1717 (2012).
[182]D. Theis, J. Ivanic, T. L. Windus, and K. Ruedenberg, J. Chem. Phys. **144**, 104304 (2016).
[183]K. Ruedenberg, M. W. Schmidt, M. M. Gilbert, and S. T. Elbert, Chem. Phys. **71**, 41 (1982).
[184]K. Ruedenberg, M. W. Schmidt, and M. M. Gilbert, Chem. Phys. **71**, 51 (1982).
[185]K. Ruedenberg, M. W. Schmidt, M. M. Gilbert, and S. T. Elbert, Chem. Phys. **71**, 65 (1982).
[186]C. Edmiston and K. Ruedenberg, Rev. Mod. Phys. **35**, 457 (1963).
[187]A. C. West, M. W. Schmidt, M. S. Gordon, and K. Ruedenberg, J. Phys. Chem. A **119**, 10360 (2015).
[188]W. C. Lu, C. Z. Wang, M. W. Schmidt, L. Bytautas, K. M. Ho, and K. Ruedenberg, J. Chem. Phys. **120**, 2638 (2004).
[189]J. Ivanic, G. J. Atchity, and K. Ruedenberg, Theor. Chem. Acc. **120**, 281 (2008).
[190]J. Ivanic and K. Ruedenberg, Theor. Chem. Acc. **120**, 295 (2008).
[191]A. C. West, M. W. Schmidt, M. S. Gordon, and K. Ruedenberg, J. Phys. Chem. A **119**, 10368 (2015).
[192]A. C. West, M. W. Schmidt, M. S. Gordon, and K. Ruedenberg, J. Phys. Chem. A **119**, 10376 (2015).
[193]A. C. West, J. J. Duchimaza-Heredia, M. S. Gordon, and K. Ruedenberg, J. Phys. Chem. A **121**, 8884 (2017).
[194]G. Schoendorff, A. C. West, M. W. Schmidt, K. Ruedenberg, A. K. Wilson, and M. S. Gordon, J. Phys. Chem. A **121**, 3588 (2017).
[195]J. J. Duchimaza Heredia, K. Ruedenberg, and M. S. Gordon, J. Phys. Chem. A **122**, 3442 (2018).
[196]J. J. Duchimaza Heredia, A. D. Sadow, and M. S. Gordon, J. Phys. Chem. A **122**, 9653 (2018).
[197]G. Schoendorff, A. C. West, M. W. Schmidt, K. Ruedenberg, and M. S. Gordon, J. Phys. Chem. A **123**, 5242 (2019).
[198]G. Schoendorff, M. W. Schmidt, K. Ruedenberg, and M. S. Gordon, J. Phys. Chem. A **123**, 5249 (2019).
[199]M. W. Schmidt, J. Ivanic, and K. Ruedenberg, J. Chem. Phys. **140**, 204104 (2014).
[200]A. C. West, M. W. Schmidt, M. S. Gordon, and K. Ruedenberg, J. Phys. Chem. A **121**, 1086 (2017).
[201]M. W. Schmidt and M. S. Gordon, Annu. Rev. Phys. Chem. **49**, 233 (1998).
[202]A. I. Krylov, Chem. Phys. Lett. **350**, 522 (2001).
[203]A. I. Krylov, Chem. Phys. Lett. **338**, 375 (2001).
[204]A. I. Krylov, Acc. Chem. Res. **39**, 83 (2006).
[205]J. S. Sears, C. D. Sherrill, and A. I. Krylov, J. Chem. Phys. **118**, 9084 (2003).
[206]D. Casanova and M. Head-Gordon, J. Chem. Phys. **129**, 064104 (2008).
[207]Y. Shao, M. Head-Gordon, and A. I. Krylov, J. Chem. Phys. **118**, 4807 (2003).

[208]D. Casanova, L. V. Slipchenko, A. I. Krylov, and M. Head-Gordon, J. Chem. Phys. **130**, 044103 (2009).
[209]S. V. Levchenko and A. I. Krylov, J. Chem. Phys. **120**, 175 (2004).
[210]J. Tomasi, B. Mennucci, and R. Cammi, Chem. Rev. **105**, 2999 (2005).
[211]M. S. Gordon, J. M. Mullin, S. R. Pruitt, L. B. Roskop, L. V. Slipchenko, and J. A. Boatz, J. Phys. Chem. B **113**, 9646 (2009).
[212]N. Minezawa and M. S. Gordon, J. Phys. Chem. A **115**, 7901 (2011).
[213]N. Minezawa and M. S. Gordon, J. Chem. Phys. **137**, 034116 (2012).
[214]D. Casanova and M. Head-Gordon, Phys. Chem. Chem. Phys. **11**, 9779 (2009).
[215]X. Zhang and J. M. Herbert, J. Chem. Phys. **143**, 234107 (2015).
[216]J. Mato and M. S. Gordon, J. Phys. Chem. A **123**, 1260 (2019).
[217]J. Mato and M. S. Gordon, Phys. Chem. Chem. Phys. **22**(3), 1475–84 (2020).
[218]F. Zahariev and M. S. Gordon, Mol. Phys. **117**, 1532 (2019).
[219]B. M. Austin, D. Y. Zubarev, and W. A. Lester, Jr., Chem. Rev. **112**, 263 (2012).
[220]M. A. Morales, J. McMinis, B. K. Clark, J. Kim, and G. E. Scuseria, J. Chem. Theory Comput. **8**, 2181 (2012).
[221]W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, Rev. Mod. Phys. **73**, 33 (2001).
[222]M. Caffarel, T. Applencourt, E. Giner, and A. Scemama, J. Chem. Phys. **144**, 151103 (2016).
[223]K. P. Esler, J. Kim, D. M. Ceperley, W. Purwanto, E. J. Walter, H. Krakauer, S. Zhang, P. R. C. Kent, R. G. Hennig, C. Umrigar *et al.*, J. Phys.: Conf. Ser. **125**, 012057 (2008).
[224]A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, and G. Hautier *et al.*, Concurrency Comput.: Pract. Exper. **27**, 5037 (2015).
[225]MongoDB, 2020.
[226]D. G. Fedorov and K. Kitaura, Chem. Phys. Lett. **597**, 99 (2014).
[227]M. Burkatzki, C. Filippi, and M. Dolg, J. Chem. Phys. **126**, 234105 (2007).
[228]V. E. Van Doren, C. Van Alsenoy, and P. Geerlings, editors, "Density Functional Theory and Its Application to Materials," Antwerp, Belgium, 8-10 June 2000, AIP Conference Proceedings, Vol. 577 (American Institute of Physics, 2001).
[229]S. S. Leang, F. Zahariev, and M. S. Gordon, J. Chem. Phys. **136**, 104101 (2012).
[230]R. Peverati and D. G. Truhlar, J. Phys. Chem. Lett. **2**, 2810 (2011).
[231]R. Peverati and D. G. Truhlar, Phys. Chem. Chem. Phys. **14**, 13171 (2012).
[232]R. Peverati and D. G. Truhlar, Phys. Chem. Chem. Phys. **14**, 16187 (2012).
[233]H. S. Yu, X. He, S. L. Li, and D. G. Truhlar, Chem. Sci. **7**, 5032 (2016).
[234]H. S. Yu, X. He, and D. G. Truhlar, J. Chem. Theory Comput. **12**, 1280 (2016).
[235]Y. Wang, P. Verma, X. Jin, D. G. Truhlar, and X. He, Proc. Natl. Acad. Sci. U. S. A. **115**, 10257 (2018).
[236]Y. Wang, X. Jin, H. S. Yu, D. G. Truhlar, and X. He, Proc. Natl. Acad. Sci. U. S. A. **114**, 8487 (2017).
[237]P. Verma, Y. Wang, S. Ghosh, X. He, and D. G. Truhlar, J. Phys. Chem. A **123**, 2966 (2019).
[238]K. R. Brorsen, N. Minezawa, F. Xu, T. L. Windus, and M. S. Gordon, J. Chem. Theory Comput. **8**, 5008 (2012).
[239]Q. Cui and M. Elstner, Phys. Chem. Chem. Phys. **16**, 14368 (2014).
[240]D. Porezag, T. Frauenheim, T. Köhler, G. Seifert, and R. Kaschner, Phys. Rev. B **51**, 12947 (1995).
[241]M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, T. Frauenheim, S. Suhai, and G. Seifert, Phys. Rev. B **58**, 7260 (1998).
[242]M. Gaus, Q. Cui, and M. Elstner, J. Chem. Theory Comput. **7**, 931 (2011).
[243]C. Köhler, G. Seifert, U. Gerstmann, M. Elstner, H. Overhof, and T. Frauenheim, Phys. Chem. Chem. Phys. **3**, 5109 (2001).
[244]V. Lutsker, B. Aradi, and T. A. Niehaus, J. Chem. Phys. **143**, 184107 (2015).
[245]Y. Nishimoto and S. Irle, Chem. Phys. Lett. **667**, 317 (2017).
[246]Y. Nishimoto, J. Phys. Chem. A **120**, 771 (2016).
[247]Y. Nishimoto, D. G. Fedorov, and S. Irle, J. Chem. Theory Comput. **10**, 4801 (2014).
[248]Y. Nishimoto, D. G. Fedorov, and S. Irle, Chem. Phys. Lett. **636**, 90 (2015).
[249]V. Q. Vuong, Y. Nishimoto, D. G. Fedorov, B. G. Sumpter, T. A. Niehaus, and S. Irle, J. Chem. Theory Comput. **15**, 3008 (2019).
[250]Y. Nishimoto, H. Nakata, D. G. Fedorov, and S. Irle, J. Phys. Chem. Lett. **6**, 5034 (2015).
[251]Y. Nishimoto and D. G. Fedorov, J. Comput. Chem. **38**, 406 (2017).
[252]V. Q. Vuong, J. Akkarapattiakal Kuriappan, M. Kubillus, J. J. Kranz, T. Mast, T. A. Niehaus, S. Irle, and M. Elstner, J. Chem. Theory Comput. **14**, 115 (2018).
[253]O. A. Von Lilienfeld, Angew. Chem., Int. Ed. **57**, 4164 (2018).
[254]L. Shen and W. Yang, J. Chem. Theory Comput. **12**, 2017 (2016).
[255]J. Zhu, B. G. Sumpter, S. Irle *et al.*, MRS Commun. **9**, 867 (2019).
[256]J. Geertsen, M. Rittby, and R. J. Bartlett, Chem. Phys. Lett. **164**, 57 (1989).
[257]A. P. Rendell, T. J. Lee, and R. Lindh, Chem. Phys. Lett. **194**, 84 (1992).
[258]V. M. Anisimov, G. H. Bauer, K. Chadalavada, R. M. Olson, J. W. Glenski, W. T. C. Kramer, E. Aprà, and K. Kowalski, J. Chem. Theory Comput. **10**, 4307 (2014).
[259]I. A. Kaliman and A. I. Krylov, J. Comput. Chem. **38**, 842 (2017).
[260]L. Dagum and R. Menon, IEEE Comput. Sci. Eng. **5**, 46 (1998).
[261]R. Kobayashi and A. P. Rendell, Chem. Phys. Lett. **265**, 1 (1997).
[262]A. C. Scheiner, G. E. Scuseria, J. E. Rice, T. J. Lee, and H. F. Schaefer III, J. Chem. Phys. **87**, 5361 (1987).
[263]T. J. Lee and A. P. Rendell, J. Chem. Phys. **94**, 6229 (1991).
[264]M. Feyereisen, G. Fitzgerald, and A. Komornicki, Chem. Phys. Lett. **208**, 359 (1993).
[265]A. P. Rendell and T. J. Lee, J. Chem. Phys. **101**, 400 (1994).
[266]E. Epifanovsky, D. Zuev, X. Feng, K. Khistyaev, Y. Shao, and A. I. Krylov, J. Chem. Phys. **139**, 134105 (2013).
[267]C. Peng, J. A. Calvin, and E. F. Valeev, Int. J. Quantum Chem. **119**, e25894 (2019).
[268]T. Shen, Z. Zhu, I. Y. Zhang, and M. Scheffler, J. Chem. Theory Comput. **15**, 4721 (2019).
[269]A. E. DePrince III and C. D. Sherrill, J. Chem. Theory Comput. **9**, 2687 (2013).
[270]T. Nakano, T. Kaminuma, T. Sato, K. Fukuzawa, Y. Akiyama, M. Uebayasi, and K. Kitaura, Chem. Phys. Lett. **351**, 475 (2002).
[271]M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T. L. Windus, and W. A. de Jong, Comput. Phys. Commun. **181**, 1477 (2010).
[272]J. M. Turney, A. C. Simmonett, R. M. Parrish, E. G. Hohenstein, F. A. Evangelista, J. T. Fermann, B. J. Mintz, L. A. Burns, J. J. Wilke, M. L. Abrams, N. J. Russ, M. L. Leininger, C. L. Janssen, E. T. Seidl, W. D. Allen, H. F. Schaefer, R. A. King, E. F. Valeev, C. D. Sherrill, and T. D. Crawford, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **2**, 556 (2012).
[273]J. F. Stanton, J. Gauss, M. E. Harding, P. G. Szalay, J. Almlöf, A. A. Auer, R. J. Bartlett, U. Benedikt, C. Berger, D. E. Bernholdt, Y. J. Bomble, L. Cheng, O. Christiansen, M. Heckert, T. Helgaker, O. Heun, C. Huber, T.-C. Jagau, H. J. A. Jensen, D. Jonsson, P. Jørgensen, J. Jusélius, K. Klein, W. J. Lauderdale, D. A. Matthews, T. Metzroth, A. V. Mitin, L. A. Mück, J. Olsen, D. P. O'Neill, D. R. Price, E. Prochnow, C. Puzzarini, K. Ruud, F. Schiffmann, W. Schwalbach, C. Simmons, S. Stopkowicz, A. Tajti, P. R. Taylor, J. Vázquez, F. Wang, J. D. Watts, and C. van Wüllen, CFOUR, Coupled-Cluster Techniques for Computational Chemistry, a Quantum-Chemical Program Package, http://www.cfour.de.
[274]QCDB, 2020.
[275]Wikipedia, 2020.
[276]V. Sundriyal, M. Sosonkina, A. Gaenko, and Z. Zhang, J. Parallel Distrib. Comput. **73**, 1157 (2013).
[277]V. Sundriyal, M. Sosonkina, B. Westheimer, and M. S. Gordon, J. Comput. Commun. **07**, 252 (2019).
[278]V. Sundriyal, M. Sosonkina, B. Westheimer, and M. Gordon, J. Comput. Commun. **06**, 184 (2018).
[279]GitHub, 2020.
[280]S. Chacon and B. Straub, *Pro Git* (Apress, 2014).
[281]V. Driessen, URL: http://Nvie.com/Posts/a-Successful-Git-Branching-Model, 2010.
[282]Travis CI (2020).
[283]Jenkins (2020).
[284]MPICH, 2020.

[285]R. C. Whaley and A. Petitet, Software: Pract. Exper. **35**, 101 (2005).
[286]S. Browne, J. Dongarra, E. Grosse, and T. Rowan, D-Lib Magazine 1, 1995.
[287]GNU, 2020.
[288]Intel, Intel Compilers, 2020.
[289]NVIDIA/PGI, PGI Compilers and Tools, 2020.
[290]E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine *et al.*, *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting* (Springer, 2004), pp. 97–104.
[291]Intel, 2020.
[292]OpenBLAS, 2020.
[293]Intel, Intel MKL, 2020.
[294]G. M. Kurtzer, V. Sochat, and M. W. Bauer, PLoS One **12**, e0177459 (2017).
[295]GAMESS Singularity, 2020.
[296]N. Newsroom, NVIDIA Newsroom Newsroom, 2017.
[297]NVIDIA Developer, 2017.
[298]J. Nieplocha, B. Palmer, V. Tipparaju, M. Krishnan, H. Trease, and E. Aprà, Int. J. High Perform. Comput. Appl. **20**, 203 (2006).
[299]E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide* (Siam, 1999).
[300]HDF5, 2020.
[301]R. H. Dennard, F. H. Gaensslen, H.-N. Yu, V. L. Rideout, E. Bassous, and A. R. Leblanc, Proc. IEEE **87**, 668 (1999).
[302]B. Caulfield, The Official NVIDIA Blog, 2009.
[303]Top500, 2019.
[304]D. G. Tomlinson, A. Asadchev, and M. S. Gordon, J. Comput. Chem. **37**, 1274 (2016).
[305]A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory* (Dover Publications, Mineola, NY, 1996).
[306]T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic-Structure Theory: Helgaker/Molecular Electronic-Structure Theory* (John Wiley & Sons, Ltd., Chichester, UK, 2000).
[307]S. Obara and A. Saika, J. Chem. Phys. **84**, 3963 (1986).
[308]L. E. McMurchie and E. R. Davidson, J. Comput. Phys. **26**, 218 (1978).
[309]J. A. Pople and W. J. Hehre, J. Comput. Phys. **27**, 161 (1978).
[310]P. M. W. Gill and J. A. Pople, Int. J. Quantum Chem. **40**, 753 (1991).
[311]P. M. W. Gill, B. G. Johnson, and J. A. Pople, Int. J. Quantum Chem. **40**, 745 (1991).
[312]M. Dupuis, J. Rys, and H. F. King, J. Chem. Phys. **65**, 111 (1976).
[313]J. Rys, M. Dupuis, and H. F. King, J. Comput. Chem. **4**, 154 (1983).
[314]J. Zhang, J. Chem. Theory Comput. **14**, 572 (2018).
[315]K. Yasuda, J. Comput. Chem. **29**, 334 (2008).
[316]S. Reine, T. Helgaker, and R. Lindh, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **2**, 290 (2012).
[317]B. P. Pritchard and E. Chow, J. Comput. Chem. **37**, 2537 (2016).
[318]Á. Rák and G. Cserey, Chem. Phys. Lett. **622**, 92 (2015).
[319]Y. Miao and K. M. Merz, J. Chem. Theory Comput. **11**, 1449 (2015).
[320]Y. Miao and K. M. Merz, J. Chem. Theory Comput. **9**, 965 (2013).
[321]A. Asadchev and M. S. Gordon, Comput. Phys. Commun. **183**, 1563 (2012).
[322]M. Häser, J. Almlöf, and M. W. Feyereisen, Theoret. Chim. Acta **79**, 115 (1991).
[323]H. Horn, H. Weiß, M. Háser, M. Ehrig, and R. Ahlrichs, J. Comput. Chem. **12**, 1058 (1991).
[324]T. P. Hamilton and H. F. Schaefer, Chem. Phys. **150**, 163 (1991).
[325]R. Lindh, Theoret. Chim. Acta **85**, 423 (1993).
[326]R. Lindh, U. Ryu, and B. Liu, J. Chem. Phys. **95**, 5889 (1991).
[327]N. Flocke and V. Lotrich, J. Comput. Chem. **29**, 2722 (2008).
[328]E. F. Valeev, Libint: A Library for the Evaluation of Molecular Integrals of Many-Body Operators over Gaussian Functions, 2019.
[329]G. M. J. Barca and P. M. W. Gill, J. Chem. Theory Comput. **12**, 4915 (2016).
[330]G. M. J. Barca and P.-F. Loos, J. Chem. Phys. **147**, 024103 (2017).
[331]M. Head-Gordon and J. A. Pople, J. Chem. Phys. **89**, 5777 (1988).
[332]SageMath Mathematical Software System, 2019.